EPICS-TwinCAT IOC

1.0

Generated by Doxygen 1.8.4

Thu Aug 1 2013 16:39:15

Contents

1 Module Index **1**

1.1 Modules . . . 1

2 Namespace Index **3**

2.1 Namespace List . . . 3

3 Hierarchical Index **5**

3.1 Class Hierarchy . . . 5

4 Class Index **7**

4.1 Class List . . . 7

5 File Index **11**

5.1 File List . . . 11

6 Module Documentation **13**

6.1 Functions called by the EPICS base . . . 13

6.1.1 Detailed Description . . . 13

6.2 OPC related functions and classes . . . 14

6.2.1 Detailed Description . . . 14

6.2.2 Typedef Documentation . . . 14

6.2.2.1 property_el . . . 14

6.2.2.2 property_map . . . 14

6.2.3 Enumeration Type Documentation . . . 14

6.2.3.1 opc_enum . . . 14

6.3 Classes for describing a structure element . . . 15

6.3.1 Detailed Description . . . 15

6.3.2 Typedef Documentation . . . 15

6.3.2.1 dimension . . . 15

6.3.2.2 dimensions . . . 15

6.3.2.3 enum_map . . . 15

6.3.2.4 enum_pair . . . 15

6.3.2.5 item_list . . . 15

6.4 Classes for describing a type . . . 16
6.4.1 Detailed Description . . . 16
6.4.2 Typedef Documentation . . . 16
6.4.2.1 type_map . . . 16
6.4.3 Enumeration Type Documentation . . . 16
6.4.3.1 type_enum . . . 16
6.5 Classes for describing a symbol . . . 17
6.5.1 Detailed Description . . . 17
6.5.2 Typedef Documentation . . . 17
6.5.2.1 symbol_list . . . 17
6.6 Classes for describing the parser . . . 18
6.6.1 Detailed Description . . . 18
6.6.2 Enumeration Type Documentation . . . 18
6.6.2.1 process_tag_enum . . . 18
6.6.2.2 process_type_enum . . . 18
6.7 XML tpy file constants . . . 20
6.7.1 Detailed Description . . . 20
6.8 OPC tpy file constants . . . 21
6.8.1 Detailed Description . . . 21
6.9 OPC property constants . . . 22
6.9.1 Detailed Description . . . 23
6.10 Utility functions and classes . . . 24
6.10.1 Detailed Description . . . 24
6.10.2 Typedef Documentation . . . 24
6.10.2.1 stringcase . . . 24
6.10.2.2 wstringcase . . . 24
6.10.3 Function Documentation . . . 25
6.10.3.1 strncasecmp . . . 25
6.10.3.2 trim_space . . . 25
6.10.3.3 trim_space . . . 25
6.10.3.4 wcsncasewcmp . . . 26
6.11 Constants related to read/write scanning . . . 27
6.11.1 Detailed Description . . . 27
6.12 Classes for describing TC symbol . . . 28
6.12.1 Detailed Description . . . 28
6.13 Classes for managing groups of TC symbols . . . 29
6.13.1 Detailed Description . . . 29
6.14 Utility functions and classes . . . 30
6.14.1 Detailed Description . . . 30
6.14.2 Enumeration Type Documentation . . . 30

6.14.2.1 case_type . . . 30
6.14.2.2 tc_epics_conv . . . 30
6.15 Classes for converting a paresed tpy . . . 31
6.15.1 Detailed Description . . . 31
6.15.2 Enumeration Type Documentation . . . 31
6.15.2.1 device_support_type . . . 31
6.15.2.2 listing_type . . . 31
6.16 EPICS maximum length constants . . . 32
6.16.1 Detailed Description . . . 32
6.17 EPICS record field names . . . 33
6.17.1 Detailed Description . . . 33
6.17.2 Variable Documentation . . . 33
6.17.2.1 EPICS_DB_ZRST . . . 33
6.17.2.2 EPICS_DB_ZRSV . . . 34
6.17.2.3 EPICS_DB_ZRVL . . . 34
6.18 Lists of EPICS record field names . . . 35
6.18.1 Detailed Description . . . 35
6.18.2 Variable Documentation . . . 35
6.18.2.1 EPICS_DB_ALLOWED . . . 35
6.18.2.2 EPICS_DB_FORBIDDEN . . . 35
6.18.2.3 EPICS_DB_NUMVAL . . . 35

7 Namespace Documentation 37
7.1 DevTc Namespace Reference . . . 37
7.1.1 Detailed Description . . . 39
7.1.2 Enumeration Type Documentation . . . 39
7.1.2.1 epics_record_enum . . . 39
7.2 EpicsTpy Namespace Reference . . . 40
7.2.1 Detailed Description . . . 41
7.3 ParseTpy Namespace Reference . . . 41
7.3.1 Detailed Description . . . 44
7.4 plc Namespace Reference . . . 44
7.4.1 Detailed Description . . . 45
7.4.2 Typedef Documentation . . . 45
7.4.2.1 BasePLCList . . . 45
7.4.2.2 BasePLCPtr . . . 45
7.4.2.3 BaseRecordList . . . 46
7.4.2.4 BaseRecordPtr . . . 46
7.4.2.5 InterfacePtr . . . 46
7.4.3 Enumeration Type Documentation . . . 46

7.4.3.1 data_type_enum . . . 46
7.5 TcComms Namespace Reference . . . 47
7.5.1 Detailed Description . . . 47
7.5.2 Function Documentation . . . 47
7.5.2.1 errorPrintf . . . 47

8 Class Documentation **49**
8.1 ParseTpy::ads_routing_info Class Reference . . . 49
8.1.1 Detailed Description . . . 50
8.1.2 Member Function Documentation . . . 50
8.1.2.1 get . . . 50
8.1.2.2 get . . . 51
8.1.2.3 set . . . 52
8.2 std::atomic< string > Class Template Reference . . . 53
8.2.1 Detailed Description . . . 54
8.3 std::atomic< wstring > Class Template Reference . . . 55
8.3.1 Detailed Description . . . 55
8.4 std::atomic_string< stringT > Class Template Reference . . . 56
8.4.1 Detailed Description . . . 57
8.4.2 Member Function Documentation . . . 57
8.4.2.1 operator= . . . 57
8.5 ParseTpy::base_record Class Reference . . . 57
8.5.1 Detailed Description . . . 59
8.5.2 Constructor & Destructor Documentation . . . 59
8.5.2.1 base_record . . . 59
8.5.2.2 base_record . . . 59
8.5.2.3 base_record . . . 60
8.5.2.4 base_record . . . 60
8.6 plc::BasePLC Class Reference . . . 60
8.6.1 Detailed Description . . . 63
8.6.2 Member Function Documentation . . . 63
8.6.2.1 add . . . 63
8.6.2.2 add . . . 63
8.6.2.3 erase . . . 64
8.6.2.4 find . . . 64
8.6.2.5 for_each . . . 65
8.6.2.6 get_next . . . 65
8.6.2.7 get_timestamp_unix . . . 65
8.6.2.8 reserve . . . 66
8.6.3 Member Data Documentation . . . 66

8.6.3.1 records . . . 66
8.7 plc::BaseRecord Class Reference . . . 66
8.7.1 Detailed Description . . . 69
8.7.2 Constructor & Destructor Documentation . . . 69
8.7.2.1 BaseRecord . . . 69
8.7.2.2 BaseRecord . . . 69
8.7.3 Member Function Documentation . . . 69
8.7.3.1 PlcPush . . . 69
8.7.3.2 PlcRead . . . 70
8.7.3.3 PlcRead . . . 71
8.7.3.4 PlcRead . . . 71
8.7.3.5 PlcReadBinary . . . 72
8.7.3.6 PlcWrite . . . 73
8.7.3.7 PlcWrite . . . 74
8.7.3.8 PlcWriteBinary . . . 74
8.7.3.9 UserPush . . . 75
8.7.3.10 UserRead . . . 76
8.7.3.11 UserRead . . . 76
8.7.3.12 UserRead . . . 77
8.7.3.13 UserReadBinary . . . 77
8.7.3.14 UserWrite . . . 78
8.7.3.15 UserWrite . . . 78
8.7.3.16 UserWrite . . . 79
8.7.3.17 UserWriteBinary . . . 79
8.8 ParseTpy::bit_location Class Reference . . . 80
8.8.1 Detailed Description . . . 81
8.9 std::case_char_traits Struct Reference . . . 81
8.9.1 Detailed Description . . . 82
8.10 std::case_wchar_traits Struct Reference . . . 82
8.10.1 Detailed Description . . . 83
8.11 TcComms::DataPar Struct Reference . . . 83
8.11.1 Detailed Description . . . 84
8.12 plc::DataValue Class Reference . . . 84
8.12.1 Detailed Description . . . 86
8.12.2 Constructor & Destructor Documentation . . . 86
8.12.2.1 DataValue . . . 86
8.12.3 Member Function Documentation . . . 87
8.12.3.1 Init . . . 87
8.12.3.2 PlcRead . . . 89
8.12.3.3 PlcRead . . . 90

8.12.3.4 PlcRead . . . 91
8.12.3.5 PlcRead . . . 91
8.12.3.6 PlcReadBinary . . . 91
8.12.3.7 PlcWrite . . . 92
8.12.3.8 PlcWrite . . . 93
8.12.3.9 PlcWrite . . . 93
8.12.3.10 PlcWrite . . . 94
8.12.3.11 PlcWriteBinary . . . 94
8.12.3.12 Read . . . 95
8.12.3.13 Read . . . 96
8.12.3.14 Read . . . 96
8.12.3.15 Read . . . 97
8.12.3.16 Read . . . 98
8.12.3.17 ReadBinary . . . 99
8.12.3.18 UserRead . . . 100
8.12.3.19 UserRead . . . 101
8.12.3.20 UserRead . . . 101
8.12.3.21 UserRead . . . 102
8.12.3.22 UserReadBinary . . . 102
8.12.3.23 UserWrite . . . 103
8.12.3.24 UserWrite . . . 104
8.12.3.25 UserWrite . . . 104
8.12.3.26 UserWrite . . . 105
8.12.3.27 UserWriteBinary . . . 105
8.12.3.28 Write . . . 106
8.12.3.29 Write . . . 107
8.12.3.30 Write . . . 107
8.12.3.31 Write . . . 108
8.12.3.32 Write . . . 108
8.12.3.33 WriteBinary . . . 109
8.12.4 Member Data Documentation . . . 110
8.12.4.1 mysize . . . 111
8.13 plc::DataValueTraits< T > Struct Template Reference . . . 111
8.13.1 Detailed Description . . . 112
8.14 plc::DataValueTypeDef Struct Reference . . . 112
8.14.1 Detailed Description . . . 115
8.15 DevTc::devTcDefIn< RecType > Struct Template Reference . . . 115
8.15.1 Detailed Description . . . 116
8.16 DevTc::devTcDefIo< RecType > Struct Template Reference . . . 116
8.16.1 Detailed Description . . . 117

8.17 DevTc::devTcDefOut< RecType > Struct Template Reference . . . 118
8.17.1 Detailed Description . . . 119
8.18 DevTc::devTcDefWaveformIn< RecType > Struct Template Reference . . . 119
8.18.1 Detailed Description . . . 120
8.19 EpicsTpy::epics_conversion Class Reference . . . 120
8.19.1 Detailed Description . . . 122
8.19.2 Constructor & Destructor Documentation . . . 122
8.19.2.1 epics_conversion . . . 122
8.19.2.2 epics_conversion . . . 122
8.19.3 Member Function Documentation . . . 122
8.19.3.1 to_epics . . . 122
8.20 EpicsTpy::epics_db_processing Class Reference . . . 124
8.20.1 Detailed Description . . . 125
8.20.2 Member Function Documentation . . . 125
8.20.2.1 operator() . . . 125
8.20.2.2 process_field_alarm . . . 129
8.20.2.3 process_field_numeric . . . 130
8.20.2.4 process_field_numeric . . . 131
8.20.2.5 process_field_string . . . 132
8.21 EpicsTpy::epics_list_processing Class Reference . . . 133
8.21.1 Detailed Description . . . 134
8.21.2 Constructor & Destructor Documentation . . . 135
8.21.2.1 epics_list_processing . . . 135
8.21.3 Member Function Documentation . . . 136
8.21.3.1 operator() . . . 136
8.22 DevTc::epics_record_traits< RecType > Struct Template Reference . . . 137
8.22.1 Detailed Description . . . 138
8.22.2 Member Data Documentation . . . 138
8.22.2.1 value_ait_type . . . 138
8.22.2.2 value_count . . . 138
8.23 DevTc::epics_record_traits< aaival > Struct Template Reference . . . 139
8.23.1 Detailed Description . . . 139
8.24 DevTc::epics_record_traits< aaoval > Struct Template Reference . . . 139
8.24.1 Detailed Description . . . 140
8.25 DevTc::epics_record_traits< airval > Struct Template Reference . . . 140
8.25.1 Detailed Description . . . 140
8.26 DevTc::epics_record_traits< aival > Struct Template Reference . . . 141
8.26.1 Detailed Description . . . 141
8.27 DevTc::epics_record_traits< aorval > Struct Template Reference . . . 141
8.27.1 Detailed Description . . . 142

8.28 DevTc::epics_record_traits< aoval > Struct Template Reference . . . 142
8.28.1 Detailed Description . . . 142
8.29 DevTc::epics_record_traits< birval > Struct Template Reference . . . 143
8.29.1 Detailed Description . . . 143
8.30 DevTc::epics_record_traits< bival > Struct Template Reference . . . 143
8.30.1 Detailed Description . . . 144
8.31 DevTc::epics_record_traits< borval > Struct Template Reference . . . 144
8.31.1 Detailed Description . . . 144
8.32 DevTc::epics_record_traits< boval > Struct Template Reference . . . 145
8.32.1 Detailed Description . . . 145
8.33 DevTc::epics_record_traits< eventval > Struct Template Reference . . . 145
8.33.1 Detailed Description . . . 146
8.34 DevTc::epics_record_traits< histogramval > Struct Template Reference . . . 146
8.34.1 Detailed Description . . . 146
8.35 DevTc::epics_record_traits< longinval > Struct Template Reference . . . 147
8.35.1 Detailed Description . . . 147
8.36 DevTc::epics_record_traits< longoutval > Struct Template Reference . . . 147
8.36.1 Detailed Description . . . 148
8.37 DevTc::epics_record_traits< mbbiDirectrval > Struct Template Reference . . . 148
8.37.1 Detailed Description . . . 148
8.38 DevTc::epics_record_traits< mbbiDirectval > Struct Template Reference . . . 149
8.38.1 Detailed Description . . . 149
8.39 DevTc::epics_record_traits< mbbirval > Struct Template Reference . . . 149
8.39.1 Detailed Description . . . 150
8.40 DevTc::epics_record_traits< mbbival > Struct Template Reference . . . 150
8.40.1 Detailed Description . . . 150
8.41 DevTc::epics_record_traits< mbboDirectrval > Struct Template Reference . . . 151
8.41.1 Detailed Description . . . 151
8.42 DevTc::epics_record_traits< mbboDirectval > Struct Template Reference . . . 151
8.42.1 Detailed Description . . . 152
8.43 DevTc::epics_record_traits< mbborval > Struct Template Reference . . . 152
8.43.1 Detailed Description . . . 152
8.44 DevTc::epics_record_traits< mbboval > Struct Template Reference . . . 153
8.44.1 Detailed Description . . . 153
8.45 DevTc::epics_record_traits< stringinval > Struct Template Reference . . . 153
8.45.1 Detailed Description . . . 154
8.46 DevTc::epics_record_traits< stringoutval > Struct Template Reference . . . 154
8.46.1 Detailed Description . . . 154
8.47 DevTc::epics_record_traits< waveformval > Struct Template Reference . . . 155
8.47.1 Detailed Description . . . 155

8.48 DevTc::epics_tc_db_processing Class Reference . . . 156
8.48.1 Detailed Description . . . 157
8.48.2 Member Function Documentation . . . 157
8.48.2.1 operator() . . . 157
8.49 DevTc::EpicsInterface Class Reference . . . 159
8.49.1 Detailed Description . . . 161
8.49.2 Member Function Documentation . . . 161
8.49.2.1 pull . . . 161
8.49.2.2 push . . . 161
8.49.3 Member Data Documentation . . . 161
8.49.3.1 callbackRequestPending . . . 161
8.49.3.2 isCallback . . . 161
8.49.3.3 isPassive . . . 162
8.50 std::std::hash< stringcase > Struct Template Reference . . . 162
8.50.1 Detailed Description . . . 162
8.50.2 Member Function Documentation . . . 162
8.50.2.1 operator() . . . 162
8.51 std::std::hash< wstringcase > Struct Template Reference . . . 162
8.51.1 Detailed Description . . . 163
8.51.2 Member Function Documentation . . . 163
8.51.2.1 operator() . . . 163
8.52 InfoInterface Class Reference . . . 163
8.52.1 Detailed Description . . . 164
8.53 InfoPLC Class Reference . . . 165
8.53.1 Detailed Description . . . 165
8.53.2 Constructor & Destructor Documentation . . . 165
8.53.2.1 InfoPLC . . . 166
8.53.2.2 ~InfoPLC . . . 166
8.54 plc::Interface Class Reference . . . 166
8.54.1 Detailed Description . . . 167
8.54.2 Constructor & Destructor Documentation . . . 168
8.54.2.1 Interface . . . 168
8.55 ParseTpy::item_record Class Reference . . . 168
8.55.1 Detailed Description . . . 169
8.56 ParseTpy::memory_location Class Reference . . . 169
8.56.1 Detailed Description . . . 171
8.56.2 Member Function Documentation . . . 171
8.56.2.1 get . . . 171
8.56.2.2 set . . . 171
8.56.2.3 set_section . . . 172

8.57 ParseTpy::opc_list Class Reference . . . 173
8.57.1 Detailed Description . . . 174
8.58 ParseTpy::parserinfo_type Class Reference . . . 174
8.58.1 Detailed Description . . . 176
8.58.2 Member Data Documentation . . . 176
8.58.2.1 name_parse . . . 176
8.59 ParseTpy::process_arg Class Reference . . . 176
8.59.1 Detailed Description . . . 178
8.59.2 Constructor & Destructor Documentation . . . 178
8.59.2.1 process_arg . . . 178
8.59.2.2 process_arg . . . 178
8.59.3 Member Function Documentation . . . 179
8.59.3.1 get . . . 179
8.59.3.2 get_full . . . 179
8.59.3.3 set . . . 180
8.60 ParseTpy::project_record Class Reference . . . 181
8.60.1 Detailed Description . . . 182
8.61 plc::scanner_thread_args Struct Reference . . . 182
8.61.1 Detailed Description . . . 182
8.62 EpicsTpy::split_io_support Class Reference . . . 183
8.62.1 Detailed Description . . . 184
8.62.2 Constructor & Destructor Documentation . . . 185
8.62.2.1 split_io_support . . . 185
8.62.3 Member Function Documentation . . . 185
8.62.3.1 increment . . . 185
8.62.3.2 operator= . . . 186
8.63 ParseTpy::symbol_record Class Reference . . . 187
8.63.1 Detailed Description . . . 188
8.64 syminfo_processing Class Reference . . . 188
8.64.1 Detailed Description . . . 189
8.65 plc::System Class Reference . . . 189
8.65.1 Detailed Description . . . 189
8.65.2 Member Function Documentation . . . 190
8.65.2.1 add . . . 190
8.65.2.2 add . . . 190
8.66 ParseTpy::tag_processing Class Reference . . . 190
8.66.1 Detailed Description . . . 191
8.66.2 Constructor & Destructor Documentation . . . 192
8.66.2.1 tag_processing . . . 192
8.66.2.2 tag_processing . . . 193

8.66.3 Member Function Documentation . . . 193
8.66.3.1 getopt . . . 193
8.67 TcComms::TCatInterface Class Reference . . . 195
8.67.1 Detailed Description . . . 198
8.67.2 Constructor & Destructor Documentation . . . 198
8.67.2.1 TCatInterface . . . 198
8.67.3 Member Function Documentation . . . 199
8.67.3.1 printTCatVal . . . 199
8.68 TcComms::TcPLC Class Reference . . . 201
8.68.1 Detailed Description . . . 203
8.68.2 Member Function Documentation . . . 203
8.68.2.1 closePort . . . 203
8.68.2.2 optimizeRequests . . . 204
8.68.2.3 set_addr . . . 206
8.69 TcComms::tcProcWrite Class Reference . . . 206
8.69.1 Detailed Description . . . 207
8.69.2 Member Function Documentation . . . 207
8.69.2.1 add . . . 207
8.69.2.2 read_ptr . . . 208
8.69.3 Member Data Documentation . . . 208
8.69.3.1 req . . . 208
8.70 DevTc::tcRegisterToIocShell Class Reference . . . 208
8.70.1 Detailed Description . . . 209
8.71 ParseTpy::tpy_file Class Reference . . . 209
8.71.1 Detailed Description . . . 210
8.71.2 Member Function Documentation . . . 211
8.71.2.1 parse_finish . . . 211
8.71.2.2 process_array . . . 211
8.71.2.3 process_symbols . . . 213
8.71.2.4 process_type_tree . . . 215
8.71.2.5 process_type_tree . . . 216
8.71.2.6 process_type_tree . . . 218
8.72 DevTc::epics_record_traits< RecType >::traits_type Struct Reference . . . 220
8.72.1 Detailed Description . . . 220
8.73 ParseTpy::type_record Class Reference . . . 220
8.73.1 Detailed Description . . . 222

9 File Documentation **223**
9.1 atomic_string.h File Reference . . . 223
9.1.1 Detailed Description . . . 224

9.2 devTc.cpp File Reference . . . 224
9.2.1 Detailed Description . . . 226
9.2.2 Function Documentation . . . 226
9.2.2.1 linkRecord . . . 226
9.3 devTc.h File Reference . . . 228
9.3.1 Detailed Description . . . 229
9.4 devTcTemplate.h File Reference . . . 229
9.4.1 Detailed Description . . . 231
9.5 drvTc.cpp File Reference . . . 232
9.5.1 Detailed Description . . . 233
9.6 drvTc.h File Reference . . . 233
9.6.1 Detailed Description . . . 234
9.7 EpicsDbGen.cpp File Reference . . . 234
9.7.1 Detailed Description . . . 234
9.7.2 Function Documentation . . . 234
9.7.2.1 main . . . 234
9.8 infoPlc.h File Reference . . . 239
9.8.1 Detailed Description . . . 239
9.9 iocMain.cpp File Reference . . . 239
9.9.1 Detailed Description . . . 240
9.10 ParseTpy.cpp File Reference . . . 240
9.10.1 Detailed Description . . . 241
9.11 ParseTpy.h File Reference . . . 241
9.11.1 Detailed Description . . . 242
9.12 ParseTpyConst.h File Reference . . . 243
9.12.1 Detailed Description . . . 245
9.13 ParseTpyInfo.cpp File Reference . . . 245
9.13.1 Detailed Description . . . 246
9.13.2 Function Documentation . . . 246
9.13.2.1 main . . . 246
9.14 ParseTpyTemplate.h File Reference . . . 248
9.14.1 Detailed Description . . . 248
9.15 plcBase.cpp File Reference . . . 248
9.15.1 Detailed Description . . . 249
9.16 plcBase.h File Reference . . . 249
9.16.1 Detailed Description . . . 251
9.17 plcBaseTemplate.h File Reference . . . 252
9.17.1 Detailed Description . . . 253
9.18 stdafx.cpp File Reference . . . 253
9.18.1 Detailed Description . . . 253

9.19 stdafx.h File Reference . . . 253
9.19.1 Detailed Description . . . 254
9.20 stringcase.cpp File Reference . . . 254
9.20.1 Detailed Description . . . 255
9.21 stringcase.h File Reference . . . 255
9.21.1 Detailed Description . . . 256
9.22 stringcase_hash.h File Reference . . . 256
9.22.1 Detailed Description . . . 257
9.23 tCat_registerRecordDeviceDriver.cpp File Reference . . . 257
9.23.1 Detailed Description . . . 259
9.24 tcComms.cpp File Reference . . . 260
9.24.1 Detailed Description . . . 260
9.25 tcComms.h File Reference . . . 261
9.25.1 Detailed Description . . . 262
9.26 TpyToEpics.cpp File Reference . . . 262
9.26.1 Detailed Description . . . 263
9.27 TpyToEpics.h File Reference . . . 263
9.27.1 Detailed Description . . . 264
9.28 TpyToEpicsConst.h File Reference . . . 264
9.28.1 Detailed Description . . . 266

Index **267**

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Functions called by the EPICS base . . . 13
OPC related functions and classes . . . 14
Classes for describing a structure element . . . 15
Classes for describing a type . . . 16
Classes for describing a symbol . . . 17
Classes for describing the parser . . . 18
XML tpy file constants . . . 20
OPC tpy file constants . . . 21
OPC property constants . . . 22
Utility functions and classes . . . 24
Constants related to read/write scanning . . . 27
Classes for describing TC symbol . . . 28
Classes for managing groups of TC symbols . . . 29
Utility functions and classes . . . 30
Classes for converting a paresed tpy . . . 31
EPICS maximum length constants . . . 32
EPICS record field names . . . 33
Lists of EPICS record field names . . . 35

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

DevTc
Namespace for TCat device support 37
EpicsTpy
Namespace for tpy-db conversion 40
ParseTpy
Namespace for tpy parsing 41
plc
Namespace for abstract plc functionality 44
TcComms
Namespace for TCat communication 47

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

ParseTpy::ads_routing_info . . . 49
- ParseTpy::project_record . . . 181

std::atomic_string< stringT > . . . 56

std::atomic_string< string > . . . 56
- std::atomic< string > . . . 53

std::atomic_string< wstring > . . . 56
- std::atomic< wstring > . . . 55

ParseTpy::base_record . . . 57
- ParseTpy::item_record . . . 168
- ParseTpy::process_arg . . . 176
- ParseTpy::symbol_record . . . 187
- ParseTpy::type_record . . . 220

plc::BasePLC . . . 60
- InfoPLC . . . 165
- TcComms::TcPLC . . . 201

ParseTpy::bit_location . . . 80
- ParseTpy::item_record . . . 168
- ParseTpy::type_record . . . 220

char_traits
- std::case_char_traits . . . 81
- std::case_wchar_traits . . . 82

TcComms::DataPar . . . 83

plc::DataValueTraits< T > . . . 111

plc::DataValueTypeDef . . . 112
- plc::BaseRecord . . . 66
- plc::DataValue . . . 84

DevTc::devTcDefIo< RecType > . . . 116
- DevTc::devTcDefIn< RecType > . . . 115
- DevTc::devTcDefOut< RecType > . . . 118
- DevTc::devTcDefWaveformIn< RecType > . . . 119

EpicsTpy::epics_conversion . . . 120
- EpicsTpy::epics_db_processing . . . 124
  - DevTc::epics_tc_db_processing . . . 156
- EpicsTpy::epics_list_processing . . . 133

DevTc::epics_record_traits< RecType > . . . 137

DevTc::epics_record_traits< aaival > . . . 139
DevTc::epics_record_traits< aaoval > . . . 139
DevTc::epics_record_traits< airval > . . . 140
DevTc::epics_record_traits< aival > . . . 141
DevTc::epics_record_traits< aorval > . . . 141
DevTc::epics_record_traits< aoval > . . . 142
DevTc::epics_record_traits< birval > . . . 143
DevTc::epics_record_traits< bival > . . . 143
DevTc::epics_record_traits< borval > . . . 144
DevTc::epics_record_traits< boval > . . . 145
DevTc::epics_record_traits< eventval > . . . 145
DevTc::epics_record_traits< histogramval > . . . 146
DevTc::epics_record_traits< longinval > . . . 147
DevTc::epics_record_traits< longoutval > . . . 147
DevTc::epics_record_traits< mbbiDirectrval > . . . 148
DevTc::epics_record_traits< mbbiDirectval > . . . 149
DevTc::epics_record_traits< mbbirval > . . . 149
DevTc::epics_record_traits< mbbival > . . . 150
DevTc::epics_record_traits< mbboDirectrval > . . . 151
DevTc::epics_record_traits< mbboDirectval > . . . 151
DevTc::epics_record_traits< mbborval > . . . 152
DevTc::epics_record_traits< mbboval > . . . 153
DevTc::epics_record_traits< stringinval > . . . 153
DevTc::epics_record_traits< stringoutval > . . . 154
DevTc::epics_record_traits< waveformval > . . . 155
std::std::hash< stringcase > . . . 162
std::std::hash< wstringcase > . . . 162
plc::Interface . . . 166
  DevTc::EpicsInterface . . . 159
  InfoInterface . . . 163
  TcComms::TCatInterface . . . 195
ParseTpy::memory_location . . . 169
  ParseTpy::process_arg . . . 176
  ParseTpy::symbol_record . . . 187
ParseTpy::opc_list . . . 173
ParseTpy::parserinfo_type . . . 174
plc::scanner_thread_args . . . 182
EpicsTpy::split_io_support . . . 183
  EpicsTpy::epics_db_processing . . . 124
  EpicsTpy::epics_list_processing . . . 133
syminfo_processing . . . 188
plc::System . . . 189
ParseTpy::tag_processing . . . 190
  ParseTpy::tpy_file . . . 209
TcComms::tcProcWrite . . . 206
DevTc::tcRegisterToIocShell . . . 208
DevTc::epics_record_traits< RecType >::traits_type . . . 220

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

ParseTpy::ads_routing_info 49
std::atomic< string >
Atomic<string> 53
std::atomic< wstring >
Atomic<wstring> 55
std::atomic_string< stringT >
Atomic strings 56
ParseTpy::base_record 57
plc::BasePLC
Base PLC 60
plc::BaseRecord
Class for managing a tag/channel 66
ParseTpy::bit_location 80
std::case_char_traits
Case insensitive traits 81
std::case_wchar_traits
Case insensitive unicode traits 82
TcComms::DataPar
Memory location struct 83
plc::DataValue
Data value 84
plc::DataValueTraits< T >
Data value traits 111
plc::DataValueTypeDef
Collection of type definitions 112
DevTc::devTcDefIn< RecType >
Device support input record 115
DevTc::devTcDefIo< RecType >
Device support record 116
DevTc::devTcDefOut< RecType >
Device support output record 118
DevTc::devTcDefWaveformIn< RecType >
Device support waveform record 119
EpicsTpy::epics_conversion 120
EpicsTpy::epics_db_processing 124
EpicsTpy::epics_list_processing 133
DevTc::epics_record_traits< RecType >
Epics record traits 137

DevTc::epics_record_traits< aaival >
Epics traits class specialization for aai record . . . 139
DevTc::epics_record_traits< aaoval >
Epics traits class specialization for aao record . . . 139
DevTc::epics_record_traits< airval >
Epics traits class specialization for ai raw record . . . 140
DevTc::epics_record_traits< aival >
Epics traits class specialization for ai record . . . 141
DevTc::epics_record_traits< aorval >
Epics traits class specialization for ao raw record . . . 141
DevTc::epics_record_traits< aoval >
Epics traits class specialization for ao record . . . 142
DevTc::epics_record_traits< birval >
Epics traits class specialization for bi raw record . . . 143
DevTc::epics_record_traits< bival >
Epics traits class specialization for bi record . . . 143
DevTc::epics_record_traits< borval >
Epics traits class specialization for bo raw record . . . 144
DevTc::epics_record_traits< boval >
Epics traits class specialization for bo record . . . 145
DevTc::epics_record_traits< eventval >
Epics traits class specialization for event record . . . 145
DevTc::epics_record_traits< histogramval >
Epics traits class specialization for histogram record . . . 146
DevTc::epics_record_traits< longinval >
Epics traits class specialization for longin record . . . 147
DevTc::epics_record_traits< longoutval >
Epics traits class specialization for longout record . . . 147
DevTc::epics_record_traits< mbbiDirectrval >
Epics traits class specialization for mbbiDirect raw record . . . 148
DevTc::epics_record_traits< mbbiDirectval >
Epics traits class specialization for mbbiDirect record . . . 149
DevTc::epics_record_traits< mbbirval >
Epics traits class specialization for mbbi raw record . . . 149
DevTc::epics_record_traits< mbbival >
Epics traits class specialization for mbbi record . . . 150
DevTc::epics_record_traits< mbboDirectrval >
Epics traits class specialization for mbboDirect raw record . . . 151
DevTc::epics_record_traits< mbboDirectval >
Epics traits class specialization for mbboDirect record . . . 151
DevTc::epics_record_traits< mbborval >
Epics traits class specialization for mbbo raw record . . . 152
DevTc::epics_record_traits< mbboval >
Epics traits class specialization for mbbo record . . . 153
DevTc::epics_record_traits< stringinval >
Epics traits class specialization for stringin record . . . 153
DevTc::epics_record_traits< stringoutval >
Epics traits class specialization for stringout record . . . 154
DevTc::epics_record_traits< waveformval >
Epics traits class specialization for waveform record . . . 155
DevTc::epics_tc_db_processing . . . 156
DevTc::EpicsInterface
Epics interface class . . . 159
std::std::hash< stringcase >
Hash for case insensitive string . . . 162
std::std::hash< wstringcase >
Hash for case insensitive unicode string . . . 162

InfoInterface
Info interface . . . 163
InfoPLC
Info plc . . . 165
plc::Interface
Abstract interface . . . 166
ParseTpy::item_record . . . 168
ParseTpy::memory_location . . . 169
ParseTpy::opc_list . . . 173
ParseTpy::parserinfo_type . . . 174
ParseTpy::process_arg . . . 176
ParseTpy::project_record . . . 181
plc::scanner_thread_args . . . 182
EpicsTpy::split_io_support . . . 183
ParseTpy::symbol_record . . . 187
syminfo_processing . . . 188
plc::System
System . . . 189
ParseTpy::tag_processing . . . 190
TcComms::TCatInterface
TCat interface class . . . 195
TcComms::TcPLC
TwinCAT PLC . . . 201
TcComms::tcProcWrite
TwinCAT process write requests . . . 206
DevTc::tcRegisterToIocShell . . . 208
ParseTpy::tpy_file . . . 209
DevTc::epics_record_traits< RecType >::traits_type
Epics record type . . . 220
ParseTpy::type_record . . . 220

Chapter 5

File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

atomic_string.h . . . 223
devTc.cpp . . . 224
devTc.h . . . 228
devTcTemplate.h . . . 229
drvTc.cpp . . . 232
drvTc.h . . . 233
EpicsDbGen.cpp . . . 234
infoPlc.h . . . 239
iocMain.cpp . . . 239
ParseTpy.cpp . . . 240
ParseTpy.h . . . 241
ParseTpyConst.h . . . 243
ParseTpyInfo.cpp . . . 245
ParseTpyTemplate.h . . . 248
plcBase.cpp . . . 248
plcBase.h . . . 249
plcBaseTemplate.h . . . 252
stdafx.cpp . . . 253
stdafx.h . . . 253
stringcase.cpp . . . 254
stringcase.h . . . 255
stringcase_hash.h . . . 256
tCat_registerRecordDeviceDriver.cpp . . . 257
tcComms.cpp . . . 260
tcComms.h . . . 261
TpyToEpics.cpp . . . 262
TpyToEpics.h . . . 263
TpyToEpicsConst.h . . . 264

Chapter 6

Module Documentation

6.1 Functions called by the EPICS base

Functions

- void **DevTc::tcLoadRecords** (const iocshArgBuf ∗args)
- void **DevTc::tcSetScanRate** (const iocshArgBuf ∗args)
- void **DevTc::tcPrintVals** (const iocshArgBuf ∗args)

6.1.1 Detailed Description

6.2 OPC related functions and classes

Classes

- class ParseTpy::opc_list
- class ParseTpy::ads_routing_info
- class ParseTpy::project_record

Typedefs

- typedef std::map< int, std::stringcase > ParseTpy::property_map
- typedef std::pair< int, std::stringcase > ParseTpy::property_el

Enumerations

- enum ParseTpy::opc_enum { ParseTpy::no_change, ParseTpy::publish, ParseTpy::silent }

6.2.1 Detailed Description

6.2.2 Typedef Documentation

6.2.2.1 typedef std::pair<int, std::stringcase> ParseTpy::property_el

This pair stores one element of opc properties.

Definition at line 36 of file ParseTpy.h.

6.2.2.2 typedef std::map<int, std::stringcase> ParseTpy::property_map

This map stores a list of opc properties.

Definition at line 32 of file ParseTpy.h.

6.2.3 Enumeration Type Documentation

6.2.3.1 enum ParseTpy::opc_enum

This enum denotes the opc state.

Enumerator

no_change Do not change inherited behaviour.

publish Publish.

silent Do not publish.

Definition at line 20 of file ParseTpy.h.

```
21 {
23     no_change,
25     publish,
27     silent
28 };
```

6.3 Classes for describing a structure element

Classes

- class ParseTpy::base_record
- class ParseTpy::bit_location
- class ParseTpy::item_record

Typedefs

- typedef std::pair< int, int > ParseTpy::dimension
- typedef std::list< dimension > ParseTpy::dimensions
- typedef std::map< int, std::stringcase > ParseTpy::enum_map
- typedef std::pair< int, std::stringcase > ParseTpy::enum_pair
- typedef std::list< item_record > ParseTpy::item_list

6.3.1 Detailed Description

6.3.2 Typedef Documentation

6.3.2.1 typedef std::pair<int, int> ParseTpy::dimension

This class stores a lbound, elements pair.

Definition at line 252 of file ParseTpy.h.

6.3.2.2 typedef std::list<dimension> ParseTpy::dimensions

This list stores lbound, elements pairs.

Definition at line 256 of file ParseTpy.h.

6.3.2.3 typedef std::map<int, std::stringcase> ParseTpy::enum_map

This map stores a list of enum values.

Definition at line 260 of file ParseTpy.h.

6.3.2.4 typedef std::pair<int, std::stringcase> ParseTpy::enum_pair

This type stores an enum pair.

Definition at line 264 of file ParseTpy.h.

6.3.2.5 typedef std::list<item_record> ParseTpy::item_list

This class stores a list of subitems.

Definition at line 277 of file ParseTpy.h.

6.4 Classes for describing a type

Classes

- class ParseTpy::type_record

Typedefs

- typedef std::map< unsigned int, type_record > ParseTpy::type_map

Enumerations

- enum ParseTpy::type_enum { ParseTpy::unknown, ParseTpy::simple, ParseTpy::arraytype, ParseTpy::enumtype, ParseTpy::structtype, ParseTpy::functionblock }

6.4.1 Detailed Description

6.4.2 Typedef Documentation

6.4.2.1 typedef std::map<unsigned int, type_record> ParseTpy::type_map

This is a map of type records, index is type number as defined in tpy

Definition at line 349 of file ParseTpy.h.

6.4.3 Enumeration Type Documentation

6.4.3.1 enum ParseTpy::type_enum

This structure describes a type record

Enumerator

unknown Unknown type.

simple Simple type.

arraytype Array type.

enumtype Enumerated type.

structtype Structure type.

functionblock Function block.

Definition at line 287 of file ParseTpy.h.

```
288 {
290     unknown,
292     simple,
294     arraytype,
296     enumtype,
298     structtype,
300     functionblock
301 };
```

6.5 Classes for describing a symbol

Classes

- class ParseTpy::memory_location
- class ParseTpy::symbol_record

Typedefs

- typedef std::list< symbol_record > ParseTpy::symbol_list

6.5.1 Detailed Description

6.5.2 Typedef Documentation

6.5.2.1 typedef std::list<symbol_record> ParseTpy::symbol_list

This is a list of symbol records

Definition at line 426 of file ParseTpy.h.

6.6 Classes for describing the parser

Classes

- class ParseTpy::process_arg
- class ParseTpy::tag_processing
- class ParseTpy::tpy_file

Enumerations

- enum ParseTpy::process_type_enum { ParseTpy::pt_invalid, ParseTpy::pt_int, ParseTpy::pt_real, ParseTpy::pt_bool, ParseTpy::pt_string, ParseTpy::pt_enum, ParseTpy::pt_binary }
- enum ParseTpy::process_tag_enum { ParseTpy::process_all, ParseTpy::process_atomic, ParseTpy::process_structured }

6.6.1 Detailed Description

6.6.2 Enumeration Type Documentation

6.6.2.1 enum ParseTpy::process_tag_enum

Enumerated type to describe the tag processing

Enumerator

process_all Process all data types.

process_atomic Process atomic data types.

process_structured Process structured data type (array, struct, function block)

Definition at line 531 of file ParseTpy.h.

```
532 {
534     process_all,
536     process_atomic,
538     process_structured
539 };
```

6.6.2.2 enum ParseTpy::process_type_enum

Enumerated type to describe the process type

Enumerator

pt_invalid Invalid type.

pt_int Numeral type.

pt_real Floating point type.

pt_bool Logic type.

pt_string String type.

pt_enum Enumerated type.

pt_binary Binary type.

Definition at line 436 of file ParseTpy.h.

```
{
    pt_invalid,
    pt_int,
    pt_real,
    pt_bool,
    pt_string,
    pt_enum,
    pt_binary
};
```

6.7 XML tpy file constants

Variables

- const char ∗const **ParseTpy::xmlPlcProjectInfo** = "PlcProjectInfo"
- const char ∗const **ParseTpy::xmlProjectInfo** = "ProjectInfo"
- const char ∗const **ParseTpy::xmlRoutingInfo** = "RoutingInfo"
- const char ∗const **ParseTpy::xmlAdsInfo** = "AdsInfo"
- const char ∗const **ParseTpy::xmlDataTypes** = "DataTypes"
- const char ∗const **ParseTpy::xmlDataType** = "DataType"
- const char ∗const **ParseTpy::xmlSymbols** = "Symbols"
- const char ∗const **ParseTpy::xmlSymbol** = "Symbol"
- const char ∗const **ParseTpy::xmlProperties** = "Properties"
- const char ∗const **ParseTpy::xmlProperty** = "Property"
- const char ∗const **ParseTpy::xmlNetId** = "NetId"
- const char ∗const **ParseTpy::xmlPort** = "Port"
- const char ∗const **ParseTpy::xmlName** = "Name"
- const char ∗const **ParseTpy::xmlType** = "Type"
- const char ∗const **ParseTpy::xmlAttrDecoration** = "Decoration"
- const char ∗const **ParseTpy::xmlIGroup** = "IGroup"
- const char ∗const **ParseTpy::xmlIOffset** = "IOffset"
- const char ∗const **ParseTpy::xmlBitSize** = "BitSize"
- const char ∗const **ParseTpy::xmlBitOffs** = "BitOffs"
- const char ∗const **ParseTpy::xmlArrayInfo** = "ArrayInfo"
- const char ∗const **ParseTpy::xmlArrayLBound** = "LBound"
- const char ∗const **ParseTpy::xmlArrayElements** = "Elements"
- const char ∗const **ParseTpy::xmlSubItem** = "SubItem"
- const char ∗const **ParseTpy::xmlFbInfo** = "FbInfo"
- const char ∗const **ParseTpy::xmlEnumInfo** = "EnumInfo"
- const char ∗const **ParseTpy::xmlEnumText** = "Text"
- const char ∗const **ParseTpy::xmlEnumEnum** = "Enum"
- const char ∗const **ParseTpy::xmlValue** = "Value"
- const char ∗const **ParseTpy::xmlDesc** = "Desc"

6.7.1 Detailed Description

6.8 OPC tpy file constants

Variables

- const char ∗const **ParseTpy::opcExport** = "opc"
- const char ∗const **ParseTpy::opcProp** = "opc_prop"
- const char ∗const **ParseTpy::opcBracket** = "["

6.8.1 Detailed Description

6.9 OPC property constants

Variables

- const int **ParseTpy::OPC_PROP_CDT** = 1
- const int **ParseTpy::OPC_PROP_VALUE** = 2
- const int **ParseTpy::OPC_PROP_QUALITY** = 3
- const int **ParseTpy::OPC_PROP_TIME** = 4
- const int **ParseTpy::OPC_PROP_RIGHTS** = 5
- const int **ParseTpy::OPC_PROP_SCANRATE** = 6
- const int **ParseTpy::OPC_PROP_UNIT** = 100
- const int **ParseTpy::OPC_PROP_DESC** = 101
- const int **ParseTpy::OPC_PROP_HIEU** = 102
- const int **ParseTpy::OPC_PROP_LOEU** = 103
- const int **ParseTpy::OPC_PROP_HIRANGE** = 104
- const int **ParseTpy::OPC_PROP_LORANGE** = 105
- const int **ParseTpy::OPC_PROP_CLOSE** = 106
- const int **ParseTpy::OPC_PROP_OPEN** = 107
- const int **ParseTpy::OPC_PROP_TIMEZONE** = 108
- const int **ParseTpy::OPC_PROP_FGC** = 201
- const int **ParseTpy::OPC_PROP_BGC** = 202
- const int **ParseTpy::OPC_PROP_BLINK** = 203
- const int **ParseTpy::OPC_PROP_BMP** = 204
- const int **ParseTpy::OPC_PROP_SND** = 205
- const int **ParseTpy::OPC_PROP_HTML** = 206
- const int **ParseTpy::OPC_PROP_AVI** = 207
- const int **ParseTpy::OPC_PROP_ALMSTAT** = 300
- const int **ParseTpy::OPC_PROP_ALMHELP** = 301
- const int **ParseTpy::OPC_PROP_ALMAREAS** = 302
- const int **ParseTpy::OPC_PROP_ALMPRIMARYAREA** = 303
- const int **ParseTpy::OPC_PROP_ALMCONDITION** = 304
- const int **ParseTpy::OPC_PROP_ALMLIMIT** = 305
- const int **ParseTpy::OPC_PROP_ALMDB** = 306
- const int **ParseTpy::OPC_PROP_ALMHH** = 307
- const int **ParseTpy::OPC_PROP_ALMH** = 308
- const int **ParseTpy::OPC_PROP_ALML** = 309
- const int **ParseTpy::OPC_PROP_ALMLL** = 310
- const int **ParseTpy::OPC_PROP_ALMROC** = 311
- const int **ParseTpy::OPC_PROP_ALMDEV** = 312
- const int **ParseTpy::OPC_PROP_PREC** = 8500
- const int **ParseTpy::OPC_PROP_ZRST** = 8510
- const int **ParseTpy::OPC_PROP_FFST** = 8525
- const int **ParseTpy::OPC_PROP_RECTYPE** = 8600
- const int **ParseTpy::OPC_PROP_INOUT** = 8601
- const char ∗const **ParseTpy::OPC_PROP_INPUT** = "input"
- const char ∗const **ParseTpy::OPC_PROP_OUTPUT** = "output"
- const int **ParseTpy::OPC_PROP_TSE** = 8602
- const int **ParseTpy::OPC_PROP_PINI** = 8603
- const int **ParseTpy::OPC_PROP_DTYP** = 8604
- const int **ParseTpy::OPC_PROP_SERVER** = 8610
- const int **ParseTpy::OPC_PROP_PLCNAME** = 8611
- const int **ParseTpy::OPC_PROP_ALMOSV** = 8700
- const int **ParseTpy::OPC_PROP_ALMZSV** = 8701
- const int **ParseTpy::OPC_PROP_ALMCOSV** = 8702
- const int **ParseTpy::OPC_PROP_ALMUNSV** = 8703
- const int **ParseTpy::OPC_PROP_ALMZRSV** = 8710
- const int **ParseTpy::OPC_PROP_ALMFFSV** = 8725

6.9.1 Detailed Description

6.10 Utility functions and classes

Classes

- struct std::case_char_traits

  case insensitive traits.
- struct std::case_wchar_traits

  case insensitive unicode traits.

Typedefs

- typedef std::basic_string
  < char, case_char_traits > std::stringcase

  case insensitive string.
- typedef std::basic_string
  < wchar_t, case_wchar_traits > std::wstringcase

  case insensitive string.

Functions

- int std::strncasecmp (const char ∗s1, const char ∗s2, int n)

  case insensitive compare with maximum length
- int std::wcsncasewcmp (const wchar_t ∗s1, const wchar_t ∗s2, int n)

  case insensitive unicode compare with maximum length
- static bool **std::case_char_traits::eq** (const char_type &c1, const char_type &c2)
- static bool **std::case_char_traits::ne** (const char_type &c1, const char_type &c2)
- static bool **std::case_char_traits::lt** (const char_type &c1, const char_type &c2)
- static int **std::case_char_traits::compare** (const char_type ∗s1, const char_type ∗s2, size_t n)
- static bool **std::case_wchar_traits::eq** (const char_type &c1, const char_type &c2)
- static bool **std::case_wchar_traits::ne** (const char_type &c1, const char_type &c2)
- static bool **std::case_wchar_traits::lt** (const char_type &c1, const char_type &c2)
- static int **std::case_wchar_traits::compare** (const char_type ∗s1, const char_type ∗s2, size_t n)
- void std::trim_space (stringcase &s)
- void std::trim_space (wstringcase &s)

6.10.1 Detailed Description

6.10.2 Typedef Documentation

6.10.2.1 typedef std::basic_string<char, case_char_traits> std::stringcase

case insensitive string.

This string class is not case sensitive.

Definition at line 76 of file stringcase.h.

6.10.2.2 typedef std::basic_string<wchar_t, case_wchar_traits> std::wstringcase

case insensitive string.

This string class is not case sensitive.

Definition at line 81 of file stringcase.h.

6.10.3 Function Documentation

6.10.3.1 int std::strncasecmp (const char * *s1*, const char * *s2*, int *n*) [inline]

case insensitive compare with maximum length

Case insensitive compare.

Parameters

s1	first string
s1	second string

Returns

<0 smaller, 0 equal, >0 greater

Definition at line 22 of file stringcase.h.

```
22                                                  {
23     return _strnicmp (s1, s2, n); }
```

6.10.3.2 void std::trim_space (stringcase & *s*)

trim space on both ends.

Parameters

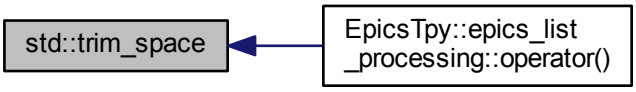
s	string to trim

Definition at line 45 of file stringcase.cpp.

Referenced by EpicsTpy::epics_list_processing::operator()().

```
46 {
47     while (!s.empty() && isspace ((unsigned char)*s.begin())) s.erase (0, 1);
48     while (!s.empty() && isspace ((unsigned char)*s.rbegin())) s.erase (s.size()-1, 1);
49 }
```

Here is the caller graph for this function:



6.10.3.3 void std::trim_space (wstringcase & *s*)

trim space on both ends.

Parameters

s	string to trim

Definition at line 53 of file stringcase.cpp.

```
54 {
55     while (!s.empty() && iswspace (*s.begin())) s.erase (0, 1);
56     while (!s.empty() && iswspace (*s.rbegin())) s.erase (s.size()-1, 1);
57 }
```

6.10.3.4 int std::wcsncasewcmp (const wchar_t * *s1*, const wchar_t * *s2*, int *n*) [inline]

case insensitive unicode compare with maximum length

Case insensitive compare for unicode string.

Parameters

s1	first string
s1	second string

Returns

<0 smaller, 0 equal, >0 greater

Definition at line 32 of file stringcase.h.

```
32                                                          {
33     return _wcsnicmp (s1, s2, n); }
```

6.11 Constants related to read/write scanning

Macros

- #define MAX_REQ_SIZE 250000

  maximum allowed request size (bytes)
- #define MAX_SINGLE_GAP_SIZE 50

  maximum allowed size (bytes) of a memory gap within continuous request
- #define MAX_REL_GAP 0.25

  (maximum allowed total gap size) / (current request size)
- #define MIN_REL_GAP_SIZE 100

  minimum allowed relative gap size (bytes)

Variables

- const int default_scanrate = 100

  default PLC TwinCAT scan rate (100ms)
- const int minimum_scanrate = 5

  minimum PLC TwinCAT scan rate (5ms)
- const int maximum_scanrate = 10000

  maximum PLC TwinCAT scan rate (10s)
- const int default_multiple = 10

  default multiple for PLC EPICS scan rate (10)
- const int minimum_multiple = 1

  minimum multiple for PLC EPICS scan rate (1)
- const int maximum_multiple = 200

  maximum multiple for PLC EPICS scan rate (200)

6.11.1 Detailed Description

6.12 Classes for describing TC symbol

Classes

- struct TcComms::DataPar

  Memory location struct.
- class TcComms::TCatInterface

  TCat interface class.

6.12.1 Detailed Description

6.13 Classes for managing groups of TC symbols

Classes

- class TcComms::tcProcWrite

  TwinCAT process write requests.
- class TcComms::TcPLC

  TwinCAT PLC.

6.13.1 Detailed Description

6.14 Utility functions and classes

Classes

- class EpicsTpy::epics_conversion
- class EpicsTpy::split_io_support

Enumerations

- enum EpicsTpy::tc_epics_conv { EpicsTpy::no_conversion, EpicsTpy::no_dot, EpicsTpy::ligo_std }
- enum EpicsTpy::case_type { EpicsTpy::preserve_case, EpicsTpy::upper_case, EpicsTpy::lower_case }

6.14.1 Detailed Description

6.14.2 Enumeration Type Documentation

6.14.2.1 enum EpicsTpy::case_type

This enum describes the case conversion rule

Enumerator

preserve_case Preserve the case.

upper_case Convert to upper case.

lower_case Convert to lower case.

Definition at line 38 of file TpyToEpics.h.

```
38                    {
40        preserve_case,
42        upper_case,
44        lower_case
45    };
```

6.14.2.2 enum EpicsTpy::tc_epics_conv

This enum describes the TwinCAT/opc to EPICS conversion rule

Enumerator

no_conversion No conversion.

no_dot Convert '.' to '_'.

ligo_std LIGO standard conversion: Eliminate leading '.', Replace sceond '.' with ':', Replace third '.' with '-', Replace all other '.' with '_'

Definition at line 23 of file TpyToEpics.h.

```
23                    {
25        no_conversion,
27        no_dot,
33        ligo_std
34    };
```

6.15 Classes for converting a paresed tpy

Classes

- class EpicsTpy::epics_list_processing
- class EpicsTpy::epics_db_processing

Enumerations

- enum EpicsTpy::listing_type { EpicsTpy::listing_standard, EpicsTpy::listing_epics, EpicsTpy::listing_autoburt }
- enum EpicsTpy::device_support_type { EpicsTpy::device_support_opc_name, EpicsTpy::device_support_tc_name }

6.15.1 Detailed Description

into an EPICS database

6.15.2 Enumeration Type Documentation

6.15.2.1 enum EpicsTpy::device_support_type

This enum describes the type of listing to produce

Enumerator

device_support_opc_name Use opc names in the INPUT/OUTPUT epics fields.

device_support_tc_name Use TwinCAT names in the INPUT/OUTPUT epics fields.

Definition at line 265 of file TpyToEpics.h.

```
265                             {
267         device_support_opc_name,
269         device_support_tc_name
270     };
```

6.15.2.2 enum EpicsTpy::listing_type

This enum describes the type of listing to produce

Enumerator

listing_standard Standard listing using TwinCAT/OPC names.

listing_epics EPICS listing using EPICS names.

listing_autoburt Autoburt listing.

Definition at line 218 of file TpyToEpics.h.

```
218                         {
220         listing_standard,
222         listing_epics,
224         listing_autoburt
225     };
```

6.16 EPICS maximum length constants

Variables

- const int **EpicsTpy::MAX_EPICS_CHANNEL** = 54
- const int **EpicsTpy::MAX_EPICS_DESC** = 40
- const int **EpicsTpy::MAX_EPICS_UNIT** = 15

6.16.1 Detailed Description

6.17 EPICS record field names

Variables

- const char ∗const **EpicsTpy::EPICS_DB_EGU** = "EGU"
- const char ∗const **EpicsTpy::EPICS_DB_DESC** = "DESC"
- const char ∗const **EpicsTpy::EPICS_DB_HOPR** = "HOPR"
- const char ∗const **EpicsTpy::EPICS_DB_LOPR** = "LOPR"
- const char ∗const **EpicsTpy::EPICS_DB_DRVH** = "DRVH"
- const char ∗const **EpicsTpy::EPICS_DB_DRVL** = "DRVL"
- const char ∗const **EpicsTpy::EPICS_DB_ONAM** = "ONAM"
- const char ∗const **EpicsTpy::EPICS_DB_ZNAM** = "ZNAM"
- const char ∗const **EpicsTpy::EPICS_DB_PREC** = "PREC"
- const char ∗const **EpicsTpy::EPICS_DB_ZRST** [16]
- const char ∗const **EpicsTpy::EPICS_DB_ZRVL** [16]
- const char ∗const **EpicsTpy::EPICS_DB_SCAN** = "SCAN"
- const char ∗const **EpicsTpy::EPICS_DB_INP** = "INP"
- const char ∗const **EpicsTpy::EPICS_DB_OUT** = "OUT"
- const char ∗const **EpicsTpy::EPICS_DB_TSE** = "TSE"
- const char ∗const **EpicsTpy::EPICS_DB_PINI** = "PINI"
- const char ∗const **EpicsTpy::EPICS_DB_DTYP** = "DTYP"
- const char ∗const **EpicsTpy::EPICS_DB_OSV** = "OSV"
- const char ∗const **EpicsTpy::EPICS_DB_ZSV** = "ZSV"
- const char ∗const **EpicsTpy::EPICS_DB_COSV** = "COSV"
- const char ∗const **EpicsTpy::EPICS_DB_HIHI** = "HIHI"
- const char ∗const **EpicsTpy::EPICS_DB_HIGH** = "HIGH"
- const char ∗const **EpicsTpy::EPICS_DB_LOW** = "LOW"
- const char ∗const **EpicsTpy::EPICS_DB_LOLO** = "LOLO"
- const char ∗const **EpicsTpy::EPICS_DB_HYST** = "HYST"
- const char ∗const **EpicsTpy::EPICS_DB_HHSV** = "HHSV"
- const char ∗const **EpicsTpy::EPICS_DB_HSV** = "HSV"
- const char ∗const **EpicsTpy::EPICS_DB_LSV** = "LSV"
- const char ∗const **EpicsTpy::EPICS_DB_LLSV** = "LLSV"
- const char ∗const **EpicsTpy::EPICS_DB_NOALARM** = "NO_ALARM"
- const char ∗const **EpicsTpy::EPICS_DB_MINOR** = "MINOR"
- const char ∗const **EpicsTpy::EPICS_DB_MAJOR** = "MAJOR"
- const char ∗const **EpicsTpy::EPICS_DB_UNSV** = "UNSV"
- const char ∗const **EpicsTpy::EPICS_DB_ZRSV** [16]

6.17.1 Detailed Description

6.17.2 Variable Documentation

6.17.2.1 const char∗ const EpicsTpy::EPICS_DB_ZRST[16]

Initial value:

```
= {"ZRST", "ONST", "TWST", "THST", "FRST", "FVST", "SXST", "SVST",
                                        "EIST", "NIST", "TEST", "ELST", "TVST", "TTST", "FTST", "FFST"}
```

Definition at line 37 of file TpyToEpicsConst.h.

6.17.2.2 const char* const EpicsTpy::EPICS_DB_ZRSV[16]

Initial value:

```
= {"ZRSV", "ONSV", "TWSV", "THSV", "FRSV", "FVSV", "SXSV", "SVSV",
                                        "EISV", "NISV", "TESV", "ELSV", "TVSV", "TTSV", "FTSV", "FFSV"}
```

Definition at line 68 of file TpyToEpicsConst.h.

6.17.2.3 const char* const EpicsTpy::EPICS_DB_ZRVL[16]

Initial value:

```
= {"ZRVL", "ONVL", "TWVL", "THVL", "FRVL", "FVVL", "SXVL", "SVVL",
                                        "EIVL", "NIVL", "TEVL", "ELVL", "TVVL", "TTVL", "FTVL", "FFVL"}
```

Definition at line 39 of file TpyToEpicsConst.h.

6.18 Lists of EPICS record field names

Variables

- const char ∗const EpicsTpy::EPICS_DB_FORBIDDEN []
- const char ∗const EpicsTpy::EPICS_DB_ALLOWED []
- const char ∗const EpicsTpy::EPICS_DB_NUMVAL []

6.18.1 Detailed Description

used to describe allowed, forbidden and numeric fields

6.18.2 Variable Documentation

6.18.2.1 const char∗ const EpicsTpy::EPICS_DB_ALLOWED[]

Initial value:

```
=   {"LINR", "EGUF", "EGUL", "AOFF", "ASLO", "ESLO", "SMOO", "HIHI", "LOLO", "HIGH", "LOW", "HHSV", "LLSV",

                                "HSV", "LSV", "HYST", "ZSV", "OSV", "COSV", "IVOA", "IVOV", "UNSV", "ZRSV",  "
      ONSV", "TWSV", "THSV",
                                "FRSV", "FVSV", "SXSV", "SVSV", "EISV", "NISV", "TESV", "ELSV", "TVSV", "TTSV"
      , "FTSV", "FFSV", NULL}
```

Names of allowed EPICS record fields

Definition at line 85 of file TpyToEpicsConst.h.

6.18.2.2 const char∗ const EpicsTpy::EPICS_DB_FORBIDDEN[]

Initial value:

```
= {"", EPICS_DB_SCAN, "DOL", EPICS_DB_INP, EPICS_DB_OUT, "VAL", "RVAL", "INIT",
                                  "ZRVL", "ONVL", "TWVL", "THVL", "FRVL", "FVVL", "SXVL", "SVVL",
                                  "EIVL", "NIVL", "TEVL", "ELVL", "TVVL", "TTVL", "FTVL", "FFVL", NULL}
```

Names of forbidden EPICS record fields

Definition at line 79 of file TpyToEpicsConst.h.

6.18.2.3 const char∗ const EpicsTpy::EPICS_DB_NUMVAL[]

Initial value:

```
= {EPICS_DB_TSE, EPICS_DB_HOPR, EPICS_DB_LOPR, EPICS_DB_DRVH, EPICS_DB_DRVL, "EGUF", "EGUL",
                                "AOFF", "ASLO", "ESLO", "SMOO", "PREC", "HIHI", "LOLO", "HIGH", "LOW", "HYST",
                                "ZRVL", "ONVL", "TWVL", "THVL", "FRVL", "FVVL", "SXVL", "SVVL", "EIVL", "NIVL"
      , "TEVL", "ELVL",
                                "TVVL", "TTVL", "FTVL", "FFVL", NULL}
```

Names of EPICS record fields which are numeric

Definition at line 91 of file TpyToEpicsConst.h.

Chapter 7

Namespace Documentation

7.1 DevTc Namespace Reference

Namespace for TCat device support.

Classes

- class EpicsInterface

  epics interface class.
- struct epics_record_traits

  epics record traits.
- struct devTcDefIo

  device support record.
- struct devTcDefIn

  device support input record.
- struct devTcDefOut

  device support output record.
- struct devTcDefWaveformIn

  device support waveform record.
- struct epics_record_traits< aaival >

  Epics traits class specialization for aai record.
- struct epics_record_traits< aaoval >

  Epics traits class specialization for aao record.
- struct epics_record_traits< aival >

  Epics traits class specialization for ai record.
- struct epics_record_traits< airval >

  Epics traits class specialization for ai raw record.
- struct epics_record_traits< aoval >

  Epics traits class specialization for ao record.
- struct epics_record_traits< aorval >

  Epics traits class specialization for ao raw record.
- struct epics_record_traits< bival >

  Epics traits class specialization for bi record.
- struct epics_record_traits< birval >

  Epics traits class specialization for bi raw record.
- struct epics_record_traits< boval >

  Epics traits class specialization for bo record.

- struct epics_record_traits< borval >

  Epics traits class specialization for bo raw record.
- struct epics_record_traits< longinval >

  Epics traits class specialization for longin record.
- struct epics_record_traits< longoutval >

  Epics traits class specialization for longout record.
- struct epics_record_traits< mbbival >

  Epics traits class specialization for mbbi record.
- struct epics_record_traits< mbbirval >

  Epics traits class specialization for mbbi raw record.
- struct epics_record_traits< mbboval >

  Epics traits class specialization for mbbo record.
- struct epics_record_traits< mbborval >

  Epics traits class specialization for mbbo raw record.
- struct epics_record_traits< mbbiDirectval >

  Epics traits class specialization for mbbiDirect record.
- struct epics_record_traits< mbbiDirectrval >

  Epics traits class specialization for mbbiDirect raw record.
- struct epics_record_traits< mbboDirectval >

  Epics traits class specialization for mbboDirect record.
- struct epics_record_traits< mbboDirectrval >

  Epics traits class specialization for mbboDirect raw record.
- struct epics_record_traits< stringinval >

  Epics traits class specialization for stringin record.
- struct epics_record_traits< stringoutval >

  Epics traits class specialization for stringout record.
- struct epics_record_traits< waveformval >

  Epics traits class specialization for waveform record.
- struct epics_record_traits< eventval >

  Epics traits class specialization for event record.
- struct epics_record_traits< histogramval >

  Epics traits class specialization for histogram record.
- class epics_tc_db_processing
- class tcRegisterToIocShell

Enumerations

- enum epics_record_enum {
  aaival = 0, aaoval, aival, aoval,
  bival, boval, eventval, histogramval,
  longinval, longoutval, mbbival, mbboval,
  mbbiDirectval, mbboDirectval, stringinval, stringoutval,
  waveformval, airval, aorval, birval,
  borval, mbbiDirectrval, mbboDirectrval, mbbirval,
  mbborval, epics_record_enumEnd, invalidval = -1 }

  epics record type enum.

Functions

- void **tcLoadRecords** (const iocshArgBuf *args)
- void **tcSetScanRate** (const iocshArgBuf *args)
- void **tcPrintVals** (const iocshArgBuf *args)

Variables

- iocshFuncDef **tcLoadRecordsFuncDef** = {"tcLoadRecords", 1, tcLoadRecordsArg}
- iocshFuncDef **tcSetScanRateFuncDef** = {"tcSetScanRate", 2, tcSetScanRateArg}
- iocshFuncDef **tcPrintValsFuncDef** = {"tcPrintVals", 1, tcPrintValsArg}

7.1.1 Detailed Description

Namespace for TCat device support. DevTc name space

DevTc namespace

7.1.2 Enumeration Type Documentation

7.1.2.1 enum DevTc::epics_record_enum

epics record type enum.

This record type enums are used as index the epics traits class

Enumerator

aaival double input array

aaoval double output array

aival double input

aoval double output

bival binary input

boval binary output

eventval event

histogramval histogram

longinval integer input

longoutval integer output

mbbival enum input

mbboval enum output

mbbiDirectval enum input direct

mbboDirectval enum output direct

stringinval string input

stringoutval string output

waveformval waveform

airval raw double input

aorval raw double output

birval raw binary input

borval raw binary output

mbbiDirectrval raw enum input

mbboDirectrval raw enum output

mbbirval raw enum input direct

mbborval raw enum output direct

epics_record_enumEnd End of enum (sentinel value)

invalidval invalid

Definition at line 105 of file devTc.h.

```
106 {
108     aaival = 0,
110     aaoval,
112     aival,
114     aoval,
116     bival,
118     boval,
120     eventval,
122     histogramval,
124     longinval,
126     longoutval,
128     mbbival,
130     mbboval,
132     mbbiDirectval,
134     mbboDirectval,
136     stringinval,
138     stringoutval,
140     waveformval,
142     airval,
144     aorval,
146     birval,
148     borval,
150     mbbiDirectrval,
152     mbboDirectrval,
154     mbbirval,
156     mbborval,
158     epics_record_enumEnd,
160     invalidval = -1
161 };
```

7.2 EpicsTpy Namespace Reference

Namespace for tpy-db conversion.

Classes

- class epics_conversion
- class split_io_support
- class epics_list_processing
- class epics_db_processing

Enumerations

- enum tc_epics_conv { no_conversion, no_dot, ligo_std }
- enum case_type { preserve_case, upper_case, lower_case }
- enum listing_type { listing_standard, listing_epics, listing_autoburt }
- enum device_support_type { device_support_opc_name, device_support_tc_name }

Variables

- const int **MAX_EPICS_CHANNEL** = 54
- const int **MAX_EPICS_DESC** = 40
- const int **MAX_EPICS_UNIT** = 15
- const char *const **EPICS_DB_EGU** = "EGU"
- const char *const **EPICS_DB_DESC** = "DESC"
- const char *const **EPICS_DB_HOPR** = "HOPR"
- const char *const **EPICS_DB_LOPR** = "LOPR"
- const char *const **EPICS_DB_DRVH** = "DRVH"
- const char *const **EPICS_DB_DRVL** = "DRVL"
- const char *const **EPICS_DB_ONAM** = "ONAM"

- const char *const **EPICS_DB_ZNAM** = "ZNAM"
- const char *const **EPICS_DB_PREC** = "PREC"
- const char *const **EPICS_DB_ZRST** [16]
- const char *const **EPICS_DB_ZRVL** [16]
- const char *const **EPICS_DB_SCAN** = "SCAN"
- const char *const **EPICS_DB_INP** = "INP"
- const char *const **EPICS_DB_OUT** = "OUT"
- const char *const **EPICS_DB_TSE** = "TSE"
- const char *const **EPICS_DB_PINI** = "PINI"
- const char *const **EPICS_DB_DTYP** = "DTYP"
- const char *const **EPICS_DB_OSV** = "OSV"
- const char *const **EPICS_DB_ZSV** = "ZSV"
- const char *const **EPICS_DB_COSV** = "COSV"
- const char *const **EPICS_DB_HIHI** = "HIHI"
- const char *const **EPICS_DB_HIGH** = "HIGH"
- const char *const **EPICS_DB_LOW** = "LOW"
- const char *const **EPICS_DB_LOLO** = "LOLO"
- const char *const **EPICS_DB_HYST** = "HYST"
- const char *const **EPICS_DB_HHSV** = "HHSV"
- const char *const **EPICS_DB_HSV** = "HSV"
- const char *const **EPICS_DB_LSV** = "LSV"
- const char *const **EPICS_DB_LLSV** = "LLSV"
- const char *const **EPICS_DB_NOALARM** = "NO_ALARM"
- const char *const **EPICS_DB_MINOR** = "MINOR"
- const char *const **EPICS_DB_MAJOR** = "MAJOR"
- const char *const **EPICS_DB_UNSV** = "UNSV"
- const char *const **EPICS_DB_ZRSV** [16]
- const char *const EPICS_DB_FORBIDDEN []
- const char *const EPICS_DB_ALLOWED []
- const char *const EPICS_DB_NUMVAL []

7.2.1 Detailed Description

Namespace for tpy-db conversion. EpicsTpy name space

7.3 ParseTpy Namespace Reference

Namespace for tpy parsing.

Classes

- class parserinfo_type
- class opc_list
- class ads_routing_info
- class project_record
- class base_record
- class bit_location
- class item_record
- class type_record
- class memory_location
- class symbol_record
- class process_arg
- class tag_processing
- class tpy_file

Typedefs

- typedef std::map< int, std::stringcase > property_map
- typedef std::pair< int, std::stringcase > property_el
- typedef std::pair< int, int > dimension
- typedef std::list< dimension > dimensions
- typedef std::map< int, std::stringcase > enum_map
- typedef std::pair< int, std::stringcase > enum_pair
- typedef std::list< item_record > item_list
- typedef std::map< unsigned int, type_record > type_map
- typedef std::list< symbol_record > symbol_list

Enumerations

- enum opc_enum { no_change, publish, silent }
- enum type_enum { unknown, simple, arraytype, enumtype, structtype, functionblock }
- enum process_type_enum { pt_invalid, pt_int, pt_real, pt_bool, pt_string, pt_enum, pt_binary }
- enum process_tag_enum { process_all, process_atomic, process_structured }

Functions

- bool **get_decoration** (const char **atts, unsigned int &decoration)

Variables

- const char *const **xmlPlcProjectInfo** = "PlcProjectInfo"
- const char *const **xmlProjectInfo** = "ProjectInfo"
- const char *const **xmlRoutingInfo** = "RoutingInfo"
- const char *const **xmlAdsInfo** = "AdsInfo"
- const char *const **xmlDataTypes** = "DataTypes"
- const char *const **xmlDataType** = "DataType"
- const char *const **xmlSymbols** = "Symbols"
- const char *const **xmlSymbol** = "Symbol"
- const char *const **xmlProperties** = "Properties"
- const char *const **xmlProperty** = "Property"
- const char *const **xmlNetId** = "NetId"
- const char *const **xmlPort** = "Port"
- const char *const **xmlName** = "Name"
- const char *const **xmlType** = "Type"
- const char *const **xmlAttrDecoration** = "Decoration"
- const char *const **xmlIGroup** = "IGroup"
- const char *const **xmlIOffset** = "IOffset"
- const char *const **xmlBitSize** = "BitSize"
- const char *const **xmlBitOffs** = "BitOffs"
- const char *const **xmlArrayInfo** = "ArrayInfo"

- const char ∗const **xmlArrayLBound** = "LBound"
- const char ∗const **xmlArrayElements** = "Elements"
- const char ∗const **xmlSubItem** = "SubItem"
- const char ∗const **xmlFbInfo** = "FbInfo"
- const char ∗const **xmlEnumInfo** = "EnumInfo"
- const char ∗const **xmlEnumText** = "Text"
- const char ∗const **xmlEnumEnum** = "Enum"
- const char ∗const **xmlValue** = "Value"
- const char ∗const **xmlDesc** = "Desc"
- const char ∗const **opcExport** = "opc"
- const char ∗const **opcProp** = "opc_prop"
- const char ∗const **opcBracket** = "["
- const int **OPC_PROP_CDT** = 1
- const int **OPC_PROP_VALUE** = 2
- const int **OPC_PROP_QUALITY** = 3
- const int **OPC_PROP_TIME** = 4
- const int **OPC_PROP_RIGHTS** = 5
- const int **OPC_PROP_SCANRATE** = 6
- const int **OPC_PROP_UNIT** = 100
- const int **OPC_PROP_DESC** = 101
- const int **OPC_PROP_HIEU** = 102
- const int **OPC_PROP_LOEU** = 103
- const int **OPC_PROP_HIRANGE** = 104
- const int **OPC_PROP_LORANGE** = 105
- const int **OPC_PROP_CLOSE** = 106
- const int **OPC_PROP_OPEN** = 107
- const int **OPC_PROP_TIMEZONE** = 108
- const int **OPC_PROP_FGC** = 201
- const int **OPC_PROP_BGC** = 202
- const int **OPC_PROP_BLINK** = 203
- const int **OPC_PROP_BMP** = 204
- const int **OPC_PROP_SND** = 205
- const int **OPC_PROP_HTML** = 206
- const int **OPC_PROP_AVI** = 207
- const int **OPC_PROP_ALMSTAT** = 300
- const int **OPC_PROP_ALMHELP** = 301
- const int **OPC_PROP_ALMAREAS** = 302
- const int **OPC_PROP_ALMPRIMARYAREA** = 303
- const int **OPC_PROP_ALMCONDITION** = 304
- const int **OPC_PROP_ALMLIMIT** = 305
- const int **OPC_PROP_ALMDB** = 306
- const int **OPC_PROP_ALMHH** = 307
- const int **OPC_PROP_ALMH** = 308
- const int **OPC_PROP_ALML** = 309
- const int **OPC_PROP_ALMLL** = 310
- const int **OPC_PROP_ALMROC** = 311
- const int **OPC_PROP_ALMDEV** = 312
- const int **OPC_PROP_PREC** = 8500
- const int **OPC_PROP_ZRST** = 8510
- const int **OPC_PROP_FFST** = 8525
- const int **OPC_PROP_RECTYPE** = 8600
- const int **OPC_PROP_INOUT** = 8601
- const char ∗const **OPC_PROP_INPUT** = "input"
- const char ∗const **OPC_PROP_OUTPUT** = "output"
- const int **OPC_PROP_TSE** = 8602

- const int **OPC_PROP_PINI** = 8603
- const int **OPC_PROP_DTYP** = 8604
- const int **OPC_PROP_SERVER** = 8610
- const int **OPC_PROP_PLCNAME** = 8611
- const int **OPC_PROP_ALMOSV** = 8700
- const int **OPC_PROP_ALMZSV** = 8701
- const int **OPC_PROP_ALMCOSV** = 8702
- const int **OPC_PROP_ALMUNSV** = 8703
- const int **OPC_PROP_ALMZRSV** = 8710
- const int **OPC_PROP_ALMFFSV** = 8725

7.3.1 Detailed Description

Namespace for tpy parsing. ParseTpy name space

7.4 plc Namespace Reference

Namespace for abstract plc functionality.

Classes

- struct scanner_thread_args
- class Interface

  abstract interface
- struct DataValueTraits

  data value traits
- struct DataValueTypeDef

  collection of type definitions
- class DataValue

  data value
- class BaseRecord

  Class for managing a tag/channel.
- class BasePLC

  base PLC
- class System

  System.

Typedefs

- typedef std::unique_ptr < Interface > InterfacePtr

  smart pointer to interface
- typedef std::shared_ptr < BaseRecord > BaseRecordPtr

  smart pointer to record
- typedef std::unordered_map < std::stringcase, BaseRecordPtr > BaseRecordList

  list of record
- typedef std::shared_ptr< BasePLC > BasePLCPtr

smart pointer to PLC

- typedef std::map < std::stringcase, BasePLCPtr > BasePLCList

  BasePLC map.

Enumerations

- enum data_type_enum {
  dtInvalid, dtBool, dtInt8, dtUInt8,
  dtInt16, dtUInt16, dtInt32, dtUInt32,
  dtInt64, dtUInt64, dtFloat, dtDouble,
  dtString, dtWString, dtBinary }

  data type enumeration

Functions

- template<>
  bool **reset_and_read** (DataValueTypeDef::atomic_bool &dirty, DataValueTypeDef::type_wstring &dest, DataValueTypeDef::atomic_string *source)
- template<>
  bool **write_and_test** (DataValueTypeDef::atomic_bool &dirty, DataValueTypeDef::atomic_bool &read_pending, DataValueTypeDef::atomic_wstring *dest, const DataValueTypeDef::type_string &source)
- VOID CALLBACK **ScannerProc** (LPVOID lpArg, DWORD dwTimerLowValue, DWORD dwTimerHighValue)
- DWORD WINAPI **scannerThread** (scanner_thread_args args)
- template<typename T , typename U >
  bool **reset_and_read** (DataValueTypeDef::atomic_bool &dirty, T &dest, U *source)
- template<typename T >
  bool **reset_and_read** (DataValueTypeDef::atomic_bool &dirty, T &dest, typename DataValueTraits< T >::traits_atomic *source)
- template<typename T , typename U >
  bool **write_and_test** (DataValueTypeDef::atomic_bool &dirty, DataValueTypeDef::atomic_bool &read_pending, U *dest, const T &source)
- template<typename T >
  bool **write_and_test** (DataValueTypeDef::atomic_bool &dirty, DataValueTypeDef::atomic_bool &read_pending, typename DataValueTraits< T >::traits_atomic *dest, const T &source)

7.4.1 Detailed Description

Namespace for abstract plc functionality. PLC namespace

7.4.2 Typedef Documentation

7.4.2.1 typedef std::map<std::stringcase, BasePLCPtr> plc::BasePLCList

BasePLC map.

This is list of BasePLC, ordered by their name.

Definition at line 755 of file plcBase.h.

7.4.2.2 typedef std::shared_ptr<BasePLC> plc::BasePLCPtr

smart pointer to PLC

This is a smart pointer to a PLC

Definition at line 750 of file plcBase.h.

7.4.2.3 typedef std::unordered_map<std::stringcase, BaseRecordPtr> plc::BaseRecordList

list of record

This is a list of tag/channel records organized as a hash map

Definition at line 589 of file plcBase.h.

7.4.2.4 typedef std::shared_ptr<BaseRecord> plc::BaseRecordPtr

smart pointer to record

This is a smart pointer to a tag/channel record

Definition at line 584 of file plcBase.h.

7.4.2.5 typedef std::unique_ptr<Interface> plc::InterfacePtr

smart pointer to interface

This is a smart pointer for Interface

Definition at line 52 of file plcBase.h.

7.4.3 Enumeration Type Documentation

7.4.3.1 enum plc::data_type_enum

data type enumeration

This is an enumerated type listing all the available data types

Enumerator

dtInvalid Invalid data type.

dtBool Boolean.

dtInt8 1-byte integer

dtUInt8 1-byte unsigned integer

dtInt16 2-byte integer

dtUInt16 2-byte unsigned integer

dtInt32 4-byte integer

dtUInt32 4-byte unsigned integer

dtInt64 8-byte integer

dtUInt64 8-byte unsigned integer

dtFloat 4-byte single precision floating point

dtDouble 8-byte double precision floating point

dtString string class

dtWString wstring class

dtBinary binary object

Definition at line 57 of file plcBase.h.

```
{
    dtInvalid,
    dtBool,
    dtInt8,
    dtUInt8,
    dtInt16,
    dtUInt16,
    dtInt32,
    dtUInt32,
    dtInt64,
    dtUInt64,
    dtFloat,
    dtDouble,
    dtString,
    dtWString,
    dtBinary
};
```

7.5 TcComms Namespace Reference

Namespace for TCat communication.

Classes

- struct DataPar

  Memory location struct.
- class TCatInterface

  TCat interface class.
- class tcProcWrite

  TwinCAT process write requests.
- class TcPLC

  TwinCAT PLC.

Typedefs

- typedef std::auto_ptr< BasePLC > **plcPtr**

Functions

- void errorPrintf (int nErr)
- bool **compByOffset** (BaseRecordPtr recA, BaseRecordPtr recB)

7.5.1 Detailed Description

Namespace for TCat communication. TcComms name space, which has all the classes and functions used for communicating with TCat.

7.5.2 Function Documentation

7.5.2.1 void TcComms::errorPrintf (int *nErr*)

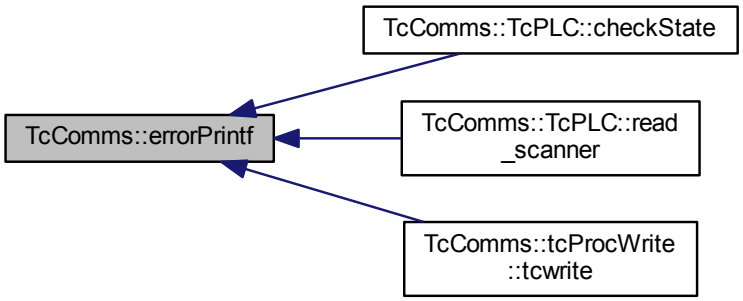
Print an error message for an ADS error return code: errorPrintf This function prints the proper error message for common ADS return codes. Documentation at: http://infosys.beckhoff.com/english.-php?content=../content/1033/tcadscommon/html/ads_returncodes.htm&id=

Definition at line 28 of file tcComms.cpp.

Referenced by TcComms::TcPLC::checkState(), TcComms::TcPLC::read_scanner(), and TcComms::tcProcWrite-::tcwrite().

```
{
    if (nErr == 6   )
        printf("target port not found, Cause: System not switched to RUN\n");
    else if (nErr == 7  )
        printf("target machine not found, Cause: Missing ADS routes\n");
    else if (nErr == 19 )
        printf("port already connected\n");
    else if (nErr == 24 )
        printf("Invalid ADS port\n");
    else if (nErr == 1285)
        printf("ROUTERERR_NOTINITIALIZED\n");
    else if (nErr == 1290)
        printf("ROUTERERR_NOTACTIVATED\n");
    else if (nErr == 1796)
        printf("reading/writing not permitted\n");
    else if (nErr == 1808)
        printf("symbol not found\n");
    else if (nErr == 1910)
        printf("server is in invalid state\n");
    else if (nErr == 1864)
        printf("ads-port not opened\n");
    else
        printf("Error code %i: ~~~MYSTERY ERROR!!!~~~\n",nErr);
}
```

Here is the caller graph for this function:



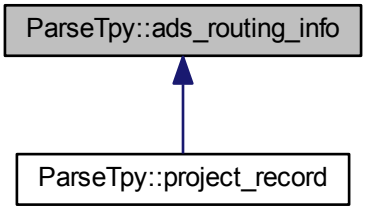
Chapter 8

Class Documentation

8.1 ParseTpy::ads_routing_info Class Reference

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::ads_routing_info:



Public Member Functions

- ads_routing_info ()

  Default constructor.
- ads_routing_info (const std::stringcase &netid, int port=801)

  Constructor.
- const std::stringcase & get_netid () const

  Get ADS net id.
- void set_netid (const std::stringcase &netid)

  Set ADS net id.
- int get_port () const

  Get ADS port.
- void set_port (int port)

  Set ADS port.
- bool isValid () const

  Checks, if net id is of the form n.n.n.n.n.n.
- std::stringcase get () const
- bool set (const std::stringcase &s)

- bool get (unsigned char &a1, unsigned char &a2, unsigned char &a3, unsigned char &a4, unsigned char &a5, unsigned char &a6) const

Protected Attributes

- std::stringcase ads_netid

  ADS net ID.
- int ads_port

  ADS port.

8.1.1 Detailed Description

This is a base class for storing the ADS routing information

Definition at line 89 of file ParseTpy.h.

8.1.2 Member Function Documentation

8.1.2.1 std::stringcase ParseTpy::ads_routing_info::get ( ) const

Gets a string representation of a ads routing information

Returns

string with format "tc://netid:port/", empty on error

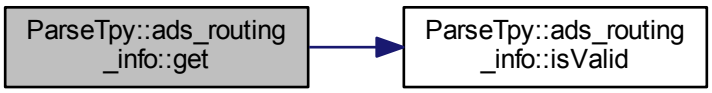
Definition at line 293 of file ParseTpy.cpp.

References ads_netid, ads_port, and isValid().

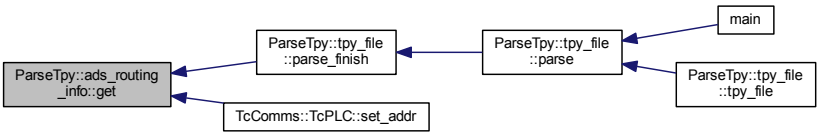
Referenced by ParseTpy::tpy_file::parse_finish(), and TcComms::TcPLC::set_addr().

```
{
    if (!isValid()) {
        return "";
    }
    char buf[256];
    sprintf_s (buf, sizeof (buf), "tc://%s:%i/", ads_netid.c_str(), ads_port);
    buf[255] = 0;
    return buf;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.1.2.2 bool ParseTpy::ads_routing_info::get (unsigned char & *a1*, unsigned char & *a2*, unsigned char & *a3*, unsigned char & *a4*, unsigned char & *a5*, unsigned char & *a6*) const

Get address in net format

Parameters

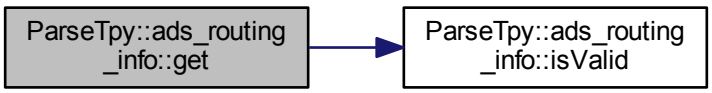
a1	First address qualifier (return)
a2	Second address qualifier (return)
a3	Third address qualifier (return)
a4	Fourth address qualifier (return)
a5	Fifth address qualifier (return)
a6	Sixth address qualifier (return)

Definition at line 306 of file ParseTpy.cpp.

References ads_netid, and isValid().

```
{
    if (!isValid()) {
        return false;
    }
    int end = 0;
    int b1(0), b2(0), b3(0), b4(0), b5(0), b6 (0);
    int num = sscanf_s (ads_netid.c_str(), "%i.%i.%i.%i.%i.%i%n",
        &b1, &b2, &b3, &b4, &b5, &b6, &end);
    if ((num == 0) || (num == EOF) || (end != ads_netid.length())) {
        return false;
    }
    a1 = b1; a2 = b2; a3 = b3;
    a4 = b4; a5 = b5; a6 = b6;
    return true;
}
```

Here is the call graph for this function:



8.1.2.3 bool ParseTpy::ads_routing_info::set (const std::stringcase & *s*)

Set the ads routing information using a string of the form: "tc://netid:port/" where netid is a string of the format n.n.n.n.n.n

Parameters

s	String describing the ads routing information

Returns

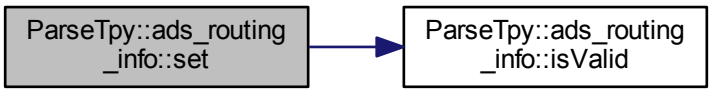
True if successful

Definition at line 327 of file ParseTpy.cpp.

References ads_netid, ads_port, and isValid().

```
{
    char buf[256];
    int p = 0;
    int end = 0;
    int num = sscanf_s (s.c_str(), " tc://%255s:%i%n",
        buf, sizeof (buf), p, &end);
    if ((num != 2) || (end != s.length())) {
        ads_netid = "";
        ads_port = 0;
        return false;
    }
    std::stringcase  ads_netid_old = ads_netid;
    int ads_port_old = ads_port;
    ads_netid = buf;
    ads_port = p;
    if (!isValid()) {
        ads_netid = ads_netid_old;
        ads_port = ads_port_old;
        return false;
    }
    return true;
}
```

Here is the call graph for this function:



The documentation for this class was generated from the following files:

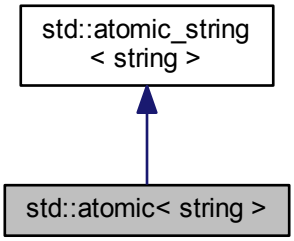
- ParseTpy.h
- ParseTpy.cpp

8.2 std::atomic< string > Class Template Reference

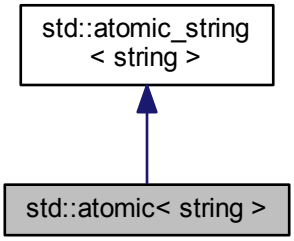
atomic<string>

```
#include <atomic_string.h>
```

Inheritance diagram for std::atomic< string >:



Collaboration diagram for std::atomic< string >:



Public Member Functions

- atomic ()

  Default constructor.
- atomic (const string &s)

  Constructor from string.

Additional Inherited Members

8.2.1 Detailed Description

template<>class std::atomic< string >

atomic<string>

This is a class implements an atomic specialization for std::string

Definition at line 71 of file atomic_string.h.

The documentation for this class was generated from the following file:

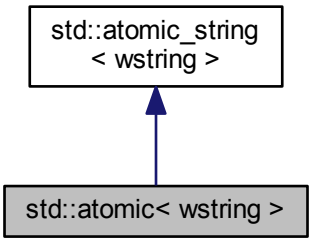
- atomic_string.h

8.3 std::atomic< wstring > Class Template Reference

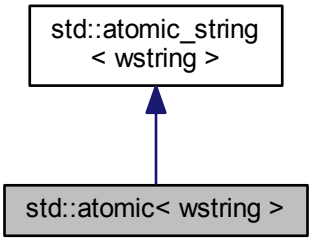
atomic<wstring>

```
#include <atomic_string.h>
```

Inheritance diagram for std::atomic< wstring >:



Collaboration diagram for std::atomic< wstring >:



Public Member Functions

- atomic ()

  Default constructor.
- atomic (const wstring &s)

  Constructor from string.

Additional Inherited Members

8.3.1 Detailed Description

template<>class std::atomic< wstring >

atomic<wstring>

This is a class implements an atomic specialization for std::wstring

Definition at line 84 of file atomic_string.h.

The documentation for this class was generated from the following file:

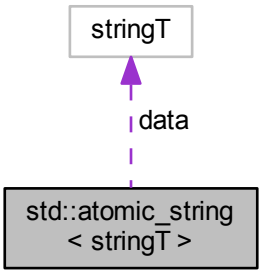
- atomic_string.h

8.4 std::atomic_string< stringT > Class Template Reference

atomic strings

```
#include <atomic_string.h>
```

Collaboration diagram for std::atomic_string< stringT >:



Public Member Functions

- atomic_string ()

  Default constructor.
- atomic_string (const stringT &s)

  Constructor from data.
- stringT operator= (const stringT &right)

  Assignment operator on basic type.
- bool is_lock_free () const

  Not lock free.
- void store (stringT value, memory_order order=memory_order_seq_cst)

  Store.
- stringT load (memory_order order=memory_order_seq_cst) const

  Load.
- operator stringT () const

  Convert to string.
- stringT exchange (const stringT &value, memory_order order=memory_order_seq_cst)

  Exchange.

- bool compare_exchange_weak (stringT &exp, const stringT &value, memory_order order1, memory_order order2)

  Compare exchange.
- bool compare_exchange_weak (stringT &exp, const stringT &value, memory_order order=memory_order_seq_cst)

  Compare exchange.
- bool compare_exchange_strong (stringT &exp, const stringT &value, memory_order order1, memory_order order2)

  Compare exchange.
- bool compare_exchange_strong (stringT &exp, const stringT &value, memory_order order=memory_order_seq_cst)

  Compare exchange.

Protected Attributes

- atomic_flag flag

  flag for spin lock
- stringT data

  data string

8.4.1 Detailed Description

template<typename stringT>class std::atomic_string< stringT >

atomic strings

This is a class implements an atomic specialization for strings

Definition at line 16 of file atomic_string.h.

8.4.2 Member Function Documentation

8.4.2.1 template<typename stringT> stringT std::atomic_string< stringT >::operator= (const stringT & *right*)

Assignment operator on basic type.

Assignment operator.

Definition at line 96 of file atomic_string.h.

```
97 {
98     while (flag.test_and_set()) {}
99     stringT ret = data;
100     data = right;
101     flag.clear();
102     return data;
103 }
```

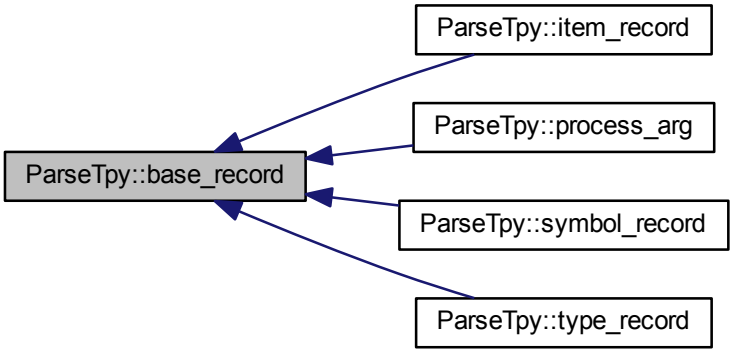
The documentation for this class was generated from the following file:

- atomic_string.h

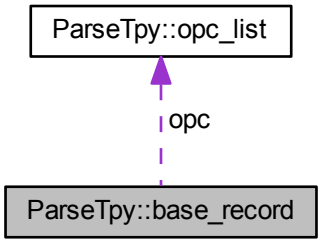
8.5 ParseTpy::base_record Class Reference

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::base_record:



Collaboration diagram for ParseTpy::base_record:



Public Member Functions

- base_record ()

  Default constructor.
- base_record (const std::stringcase &n)
- base_record (const std::stringcase &n, const opc_list &o)
- base_record (const std::stringcase &n, const opc_list &o, const std::stringcase &tn, int td=0)
- base_record (const std::stringcase &n, const std::stringcase &tn, int td=0)
- const std::stringcase & get_name () const

  Get name.
- std::stringcase & get_name ()

  Get name.
- void set_name (std::stringcase n)

  Set name.
- const std::stringcase & get_type_name () const

  Get type name.

- std::stringcase & get_type_name ()

  Get type name.
- void set_type_name (std::stringcase t)

  Set type name.
- unsigned int get_type_decoration () const

  Get type decoration.
- void set_type_decoration (unsigned int id)

  Set type decoration.
- const opc_list & get_opc () const

  Get OPC list.
- opc_list & get_opc ()

  Get OPC list.

Protected Attributes

- std::stringcase name

  name of type
- std::stringcase type_n

  type definition
- unsigned int type_decoration

  decoration or type ID of type definition
- opc_list opc

  list of opc properties

8.5.1 Detailed Description

This is a base class for storing name, type, type id and opc list

Definition at line 157 of file ParseTpy.h.

8.5.2 Constructor & Destructor Documentation

8.5.2.1 ParseTpy::base_record::base_record (const std::stringcase & *n*) [inline],[explicit]

Constructor

Parameters

n	Name

Definition at line 164 of file ParseTpy.h.

```
165         : name (n), type_decoration (0) {}
```

8.5.2.2 ParseTpy::base_record::base_record (const std::stringcase & *n*, const opc_list & *o*) [inline]

Constructor

Parameters

n	Name
o	OPC list

Definition at line 169 of file ParseTpy.h.

```
170         : name (n), opc (o), type_decoration (0) {}
```

8.5.2.3 ParseTpy::base_record::base_record (const std::stringcase & *n*, const opc_list & *o*, const std::stringcase & *tn*, int *td* = 0) [inline]

Constructor

Parameters

n	Name
o	OPC list
tn	Type name
td	Type decortation or id

Definition at line 176 of file ParseTpy.h.

```
178         : name (n), opc (o), type_n (tn), type_decoration (td) {}
```

8.5.2.4 ParseTpy::base_record::base_record (const std::stringcase & *n*, const std::stringcase & *tn*, int *td* = 0) [inline]

Constructor

Parameters

n	Name
tn	Type name
td	Type decortation or id

Definition at line 183 of file ParseTpy.h.

```
185         : name (n), type_n (tn), type_decoration (td) {}
```

The documentation for this class was generated from the following file:

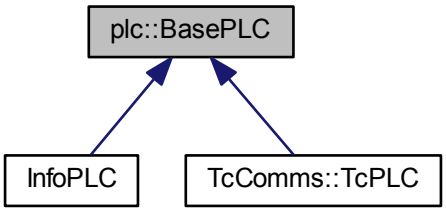
- ParseTpy.h

8.6 plc::BasePLC Class Reference

base PLC

```
#include <plcBase.h>
```

Inheritance diagram for plc::BasePLC:



Public Types

- typedef std::recursive_mutex mutex_type

  Defines the mutex type.
- typedef std::lock_guard < mutex_type > guard

  Defined the mutex guard type.
- typedef DataValueTypeDef::type_uint64 time_type

  Define timestamp type.
- typedef void(BasePLC::∗ scanner_func)()

  Function pointer to scanner.

Public Member Functions

- BasePLC ()

  Default constructor.
- virtual ∼BasePLC ()

  Destructor.
- int get_read_scanner_period () const

  Get read scannner period in ms.
- void set_read_scanner_period (int period)

  Set read scannner period in ms.
- bool start_read_scanner ()

  Start read scannner.
- bool stop_read_scanner ()

  Stop read scannner.
- bool terminate_read_scanner ()

  Terminate read scannner.
- int get_write_scanner_period () const

  Get write scannner period in ms.
- void set_write_scanner_period (int period)

  Set write scannner period in ms.
- bool start_write_scanner ()

  Start write scannner.

- bool stop_write_scanner ()

  Stop write scannner.
- bool terminate_write_scanner ()

  Terminate write scannner.
- int get_update_scanner_period () const

  Get update scannner period in ms.
- void set_update_scanner_period (int period)

  Set update scannner period in ms.
- bool start_update_scanner ()

  Start update scannner.
- bool stop_update_scanner ()

  Stop update scannner.
- bool terminate_update_scanner ()

  Terminate update scannner.
- void reserve (BaseRecordList::size_type n)
- bool add (BaseRecord *precord)
- bool add (BaseRecordPtr precord)
- BaseRecordPtr find (const std::stringcase &name)
- bool erase (const std::stringcase &name)
- bool get_next (BaseRecordPtr &next, const BaseRecordPtr &prev) const
- template<typename func >
  void for_each (func &f)
- virtual void printAllRecords ()

  Print all records and vals to stdout. (override for action)
- time_type get_timestamp () const

  Get time stamp.
- time_t get_timestamp_unix () const
- void set_timestamp (time_type tstamp)

  Set time stamp.
- void update_timestamp ()

  Set time stamp to current time.
- std::stringcase get_name () const

  Get name.
- void **test** ()
- virtual bool **start** ()

Protected Member Functions

- void set_name (const std::stringcase &n)

  Set name (careful! This is used for indexing in the PLCList of System)
- virtual void read_scanner ()

  read scanner (override for action)
- virtual void write_scanner ()

  write scanner (override for action)
- virtual void update_scanner ()

  update scanner (override for action)

Protected Attributes

- mutex_type mux

  Mutex to synchronize access to this class.
- std::stringcase name

  Name.
- BaseRecordList records
- time_type timestamp

  Time stamp.
- int read_scanner_period

  read scanner period in ms
- int write_scanner_period

  write scanner period in ms
- int update_scanner_period

  update scanner period in ms
- std::thread read_thread

  read thread
- std::thread write_thread

  write thread
- std::thread update_thread

  update thread

8.6.1 Detailed Description

base PLC

This is a base class for interfacing a programmable logic controller. It contains and manages a list of tag/channel records. This is a base class which needs to be used a derived class by a real implementation.

This class is MT safe and uses a mutex to synchronize access.

Definition at line 599 of file plcBase.h.

8.6.2 Member Function Documentation

8.6.2.1 bool plc::BasePLC::add (BaseRecord * *precord*) [inline]

Add a new tag/channel record. Adding a duplicate is not possible.

Parameters

record	Pointer to record. Will be adopted

Returns

true, if it could be added

Definition at line 664 of file plcBase.h.

```
664                                     {
665         return precord ? add (BaseRecordPtr (precord)) : false; }
```

8.6.2.2 bool plc::BasePLC::add (BaseRecordPtr *precord*)

Add a new tag/channel record. Adding a duplicate is not possible.

Parameters

record	Smart pointer to record.

Returns

true, if it could be added

Definition at line 479 of file plcBase.cpp.

References mux, and records.

```
480 {
481     if (!precord.get()) {
482         return false;
483     }
484     guard lock (mux);
485     auto ret = records.insert (
486         BaseRecordList::value_type (precord->get_name(), precord));
487     return ret.second;
488 }
```

8.6.2.3 bool plc::BasePLC::erase (const std::stringcase & *name*)

Erase a tag/channel record.

Parameters

name	Name of record

Returns

true if erased

Definition at line 506 of file plcBase.cpp.

References records.

```
507 {
508     auto num = records.erase (name);
509     return num > 0;
510 }
```

8.6.2.4 BaseRecordPtr plc::BasePLC::find (const std::stringcase & *name*)

Find a new tag/channel record.

Parameters

name	Name of record

Returns

Smart pointer to record (contains nullptr when not found)

Definition at line 492 of file plcBase.cpp.

References mux, and records.

```
493 {
494     guard lock (mux);
495     auto i = records.find (name);
496     if (i == records.end()) {
497         return BaseRecordPtr();
498     }
499     else {
500         return i->second;
501     }
502 }
```

8.6.2.5 template<typename func > void plc::BasePLC::for_each (func & *f*)

Iterate over all list elements This will yield good performance, but will lock the PLC for the entire processing time

Parameters

f	Function which takes BaseRecord∗ as the argument

Definition at line 197 of file plcBaseTemplate.h.

References mux, and records.

```
198 {
199     guard lock (mux);
200     for (auto i: records) {
201         f (i.second.get());
202     }
203 }
```

8.6.2.6 bool plc::BasePLC::get_next (BaseRecordPtr & *next*, const BaseRecordPtr & *prev*) const

Get next record in list. Resets to the beginning, if the list changes between subsequest access. This is an MT safe access method to cycle through the list. However, it is not high performance and is meant for slow external acecss.

Parameters

next	Next tag/channel record (return)
previous	tag/channel record

Returns

true if successful

Definition at line 514 of file plcBase.cpp.

References mux, and records.

```
515 {
516     guard lock (mux);
517     if (records.empty() || !prev) {
518         return false;
519     }
520     auto i = records.find (prev->get_name().c_str());
521     if (i != records.cend()) {
522         ++i;
523     }
524     if (i == records.cend()) {
525         next = records.cbegin()->second;
526     }
527     else {
528         next = i->second;
529     }
530     return true;
531 }
```

8.6.2.7 time_t plc::BasePLC::get_timestamp_unix ( ) const

Get time stamp as unix time (seconds since 1970-01-01 00:00:00) Deos not include leap seconds

Definition at line 538 of file plcBase.cpp.

References timestamp.

```
539 {
540     time_type temp;
541     //convert from 100ns intervals to seconds
542     temp = timestamp / TICKS_PER_SECOND;
543     // too early? return 0
```

```
    if (temp < EPOCH_DIFFERENCE) {
        return 0;
    }
    //subtract number of seconds between epochs
    else {
        return (time_t) (temp - EPOCH_DIFFERENCE);
    }
}
```

8.6.2.8 void plc::BasePLC::reserve (BaseRecordList::size_type *n*) [inline]

Reserves the given number of elements in the tag/channel list. Use this function when you know many elements are added beforehand to avoid unnecessary rehashing.

Parameters

n	Number of expected tag/channel records

Definition at line 659 of file plcBase.h.

References records.

```
659                                         {
660         records.reserve (n); }
```

8.6.3 Member Data Documentation

8.6.3.1 BaseRecordList plc::BasePLC::records [protected]

List of tags/channels. The load factor is initialized to 0.5.

Definition at line 723 of file plcBase.h.

Referenced by add(), BasePLC(), erase(), find(), for_each(), get_next(), TcComms::TcPLC::printAllRecords(), and reserve().

The documentation for this class was generated from the following files:

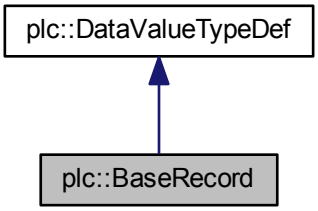
- plcBase.h
- plcBase.cpp
- plcBaseTemplate.h

8.7 plc::BaseRecord Class Reference

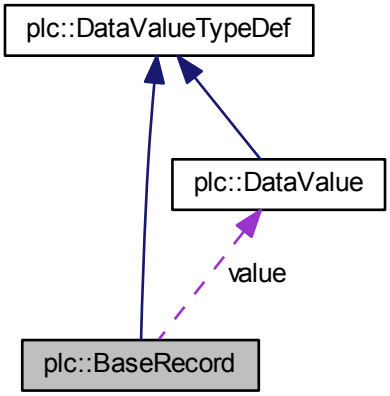
Class for managing a tag/channel.

```
#include <plcBase.h>
```

Inheritance diagram for plc::BaseRecord:



Collaboration diagram for plc::BaseRecord:



Public Member Functions

- BaseRecord ()

  Default constructor.
- BaseRecord (const std::stringcase &tag)
- BaseRecord (const std::stringcase &recordName, data_type_enum rt, Interface *puser=nullptr, Interface *pplc=nullptr)
- virtual ~BaseRecord ()

  Desctructor.
- const std::stringcase & get_name () const

  Get name.
- void set_name (const std::stringcase &recordName)

  Set name.
- bool get_process () const

  Get process flag: false = disabled, true = enabled.
- void set_process (bool isEnabled)

Set process flag.

- virtual Interface * get_userInterface () const

  Get pointer to user interface (no ownership transfer)
- virtual Interface * get_plcInterface () const

  Get pointer to plc interface (no ownership transfer)
- void set_userInterface (Interface *puser)

  Set user interface (object will be adopted!)
- void set_plcInterface (Interface *pplc)

  Set plc interface (object will be adopted!)
- const DataValue & get_data () const

  Get a const reference to the data object.
- DataValue & get_data ()

  Get a reference to the data object.
- template<typename T >
  bool UserRead (T &data)
- bool UserRead (type_string_value *data, size_type max)
- bool UserRead (DataValue::type_wstring_value *data, size_type max)
- template<typename T >
  bool UserWrite (const T &data)
- bool UserWrite (const type_string_value *data, size_type max)
- bool UserWrite (const type_wstring_value *data, size_type max)
- size_type UserReadBinary (type_binary p, size_type len)
- size_type UserWriteBinary (const type_binary p, size_type len)
- bool UserIsDirty () const

  Ckecks if the user needs to read an updated value.
- bool UserPush (bool force=false)
- bool UserPull ()

  Pulls the user for new data.
- template<typename T >
  bool PlcRead (T &data)
- bool PlcRead (type_string_value *data, size_type max)
- bool PlcRead (type_wstring_value *data, size_type max)
- template<typename T >
  bool **PlcWrite** (const T &data)
- bool PlcWrite (const type_string_value *data, size_type max)
- bool PlcWrite (const type_wstring_value *data, size_type max)
- size_type PlcReadBinary (type_binary p, size_type len)
- size_type PlcWriteBinary (const type_binary p, size_type len)
- bool PlcIsDirty () const

  Checks if the plc needs to read an updated value.
- bool PlcPush (bool force=false)
- bool PlcPull ()

  Pulls the plc for new data.

Protected Attributes

- std::stringcase name

  Name.
- unsigned long long timestamp

  Time stamp.
- bool process

  Process flag: false = disabled, true = enabled.

- DataValue value

  Data value.
- InterfacePtr plc

  PLC interface (master)
- InterfacePtr user

  User interface (slave)

Additional Inherited Members

8.7.1 Detailed Description

Class for managing a tag/channel.

This is the base class for a tag/channel. It contains a data value and owns two pointers to a user and plc interface, respectively. This is the basic work horse for exchanging data between a user (slave) and a plc (master).

Definition at line 423 of file plcBase.h.

8.7.2 Constructor & Destructor Documentation

8.7.2.1 plc::BaseRecord::BaseRecord (const std::stringcase & *tag*) [inline],[explicit]

Constructor

Parameters

tag	Name of tag/channel

Definition at line 430 of file plcBase.h.

```
431         : name (tag), process (true) {}
```

8.7.2.2 plc::BaseRecord::BaseRecord (const std::stringcase & *recordName*, data_type_enum *rt*, Interface * *puser* = nullptr, Interface * *pplc* = nullptr) [inline]

Constructor

Parameters

tag	Name of tag/channel
puser	Pointer to user interface object (will be adopted!)
puser	Pointer to plc interface object (will be adopted!)

Definition at line 436 of file plcBase.h.

```
438         : name (recordName), process (true), value (rt), user (puser),
      plc (pplc) {}
```

8.7.3 Member Function Documentation

8.7.3.1 bool plc::BaseRecord::PlcPush (bool *force* = false) [inline]

Initiated a plc pull, if the data needs an update

Parameters

force	Forces a plc update even when not needed

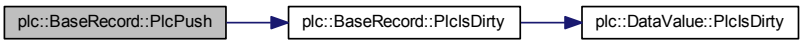
Definition at line 172 of file plcBaseTemplate.h.

References plc, and PlcIsDirty().

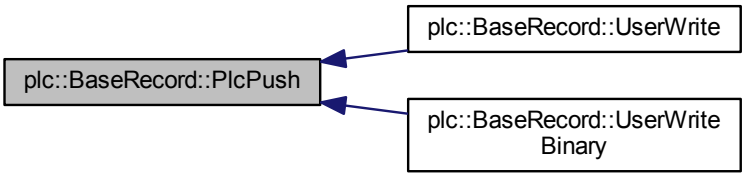
Referenced by UserWrite(), and UserWriteBinary().

```
173 {
174     if (plc.get() && (force || PlcIsDirty())) {
175         return plc->push();
176     }
177     else {
178         return false;
179     }
180 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.7.3.2 template<typename T > bool plc::BaseRecord::PlcRead (T & *data*) [inline]

Execute a plc read, but pull user first

Parameters

data	Reference to data (return)

Returns

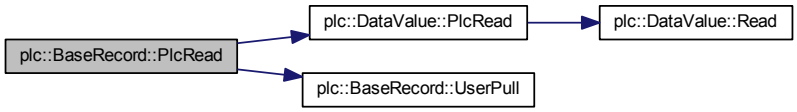
true if successfull

Definition at line 520 of file plcBase.h.

References plc::DataValue::PlcRead(), UserPull(), and value.

```
520                                                 {
521         UserPull(); return value.PlcRead (data); }
```

Here is the call graph for this function:



8.7.3.3 bool plc::BaseRecord::PlcRead (type_string_value * *data*, size_type *max*) [inline]

Execute a plc read, but pull user first

Parameters

data	character pointer, pchar (return)

Returns

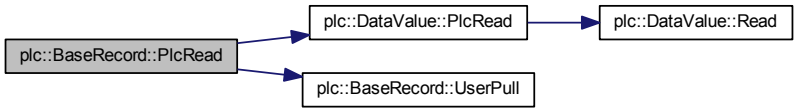
true if successfull

Definition at line 525 of file plcBase.h.

References plc::DataValue::PlcRead(), UserPull(), and value.

```
525                                                     {
526         UserPull(); return value.PlcRead (data, max); }
```

Here is the call graph for this function:



8.7.3.4 bool plc::BaseRecord::PlcRead (type_wstring_value * *data*, size_type *max*) [inline]

Execute a plc read, but pull user first

Parameters

data	character pointer, pwchar (return)

Returns

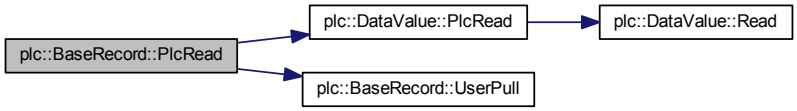
true if successfull

Definition at line 530 of file plcBase.h.

References plc::DataValue::PlcRead(), UserPull(), and value.

```
                                                    {
        UserPull(); return value.PlcRead (data, max); }
```

Here is the call graph for this function:



8.7.3.5 size_type plc::BaseRecord::PlcReadBinary (type_binary *p*, size_type *len*) [inline]

Execute a plc read, but pull user first

Parameters

p	Pointer to data (destination buffer)
len	Length in bytes (must be the same as data length)

Returns

Number of bytes read (0 on error)

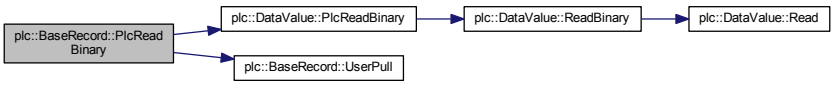
Definition at line 549 of file plcBase.h.

References plc::DataValue::PlcReadBinary(), UserPull(), and value.

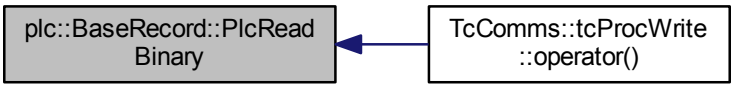
Referenced by TcComms::tcProcWrite::operator()().

```
                                                    {
        UserPull(); return value.PlcReadBinary (p, len); }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.7.3.6 bool plc::BaseRecord::PlcWrite (const type_string_value * *data,* size_type *max*) [inline]

Execute a plc write and push user

Parameters

data	character pointer, pchar

Returns

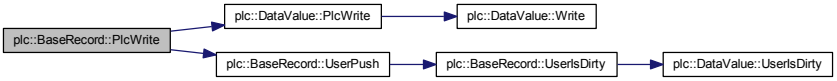
true if successfull

Definition at line 537 of file plcBase.h.

References plc::DataValue::PlcWrite(), UserPush(), and value.

```
537                                                                 {
538         bool ret = value.PlcWrite (data, max); if (ret) UserPush(); return ret; }
```

Here is the call graph for this function:



8.7.3.7 bool plc::BaseRecord::PlcWrite (const type_wstring_value * *data*, size_type *max*) [inline]

Execute a plc write and push user

Parameters

data	character pointer, pwchar

Returns

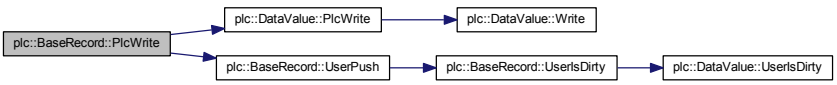
true if successfull

Definition at line 542 of file plcBase.h.

References plc::DataValue::PlcWrite(), UserPush(), and value.

```
542                                                                 {
543         bool ret = value.PlcWrite (data, max); if (ret) UserPush(); return ret; }
```

Here is the call graph for this function:



8.7.3.8 size_type plc::BaseRecord::PlcWriteBinary (const type_binary *p*, size_type *len*) [inline]

Execute a plc write and push user

Parameters

p	Pointer to data (source buffer)
len	Length in bytes (must be the same as data length)

Returns

Number of bytes written (0 on error)

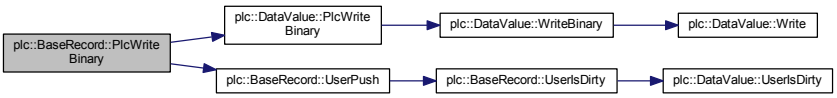
Definition at line 555 of file plcBase.h.

References plc::DataValue::PlcWriteBinary(), UserPush(), and value.

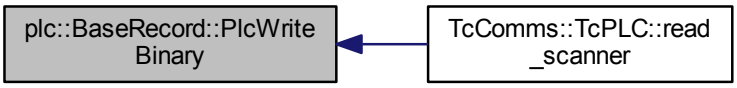
Referenced by TcComms::TcPLC::read_scanner().

```
555                                                                 {
556         size_type ret = value.PlcWriteBinary (p, len);
557         if (ret > 0) UserPush(); return ret; }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.7.3.9 bool plc::BaseRecord::UserPush (bool *force* = `false`) [inline]

Initiated a user pull, if the data needs an update

Parameters

force	Forces a user update even when not needed

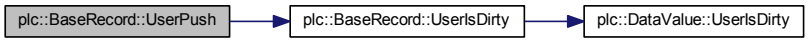
Definition at line 148 of file plcBaseTemplate.h.

References user, and UserIsDirty().

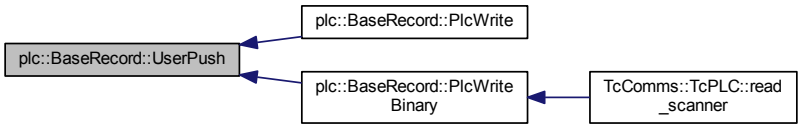
Referenced by PlcWrite(), and PlcWriteBinary().

```
149 {
150     if (user.get() && (force || UserIsDirty())) {
151         return user->push();
152     }
153     else {
154         return false;
155     }
156 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.7.3.10 template<typename T > bool plc::BaseRecord::UserRead (T & *data*) [inline]

Execute a user read, but pull plc first

Parameters

data	Reference to data (return)

Returns

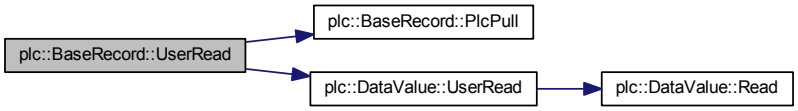
true if successfull

Definition at line 468 of file plcBase.h.

References PlcPull(), plc::DataValue::UserRead(), and value.

```
468                                                 {
469         PlcPull(); return value.UserRead (data); }
```

Here is the call graph for this function:



8.7.3.11 bool plc::BaseRecord::UserRead (type_string_value * *data*, size_type *max*) [inline]

Execute a user read, but pull plc first

Parameters

data	character pointer, pchar (return)

Returns

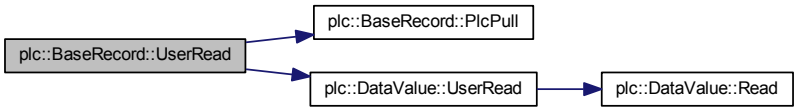
true if successfull

Definition at line 473 of file plcBase.h.

References PlcPull(), plc::DataValue::UserRead(), and value.

```
473                                                     {
474         PlcPull(); return value.UserRead (data, max); }
```

Here is the call graph for this function:



8.7.3.12 bool plc::BaseRecord::UserRead (DataValue::type_wstring_value * *data*, size_type *max*) [inline]

Execute a user read, but pull plc first

Parameters

data	character pointer, pwchar (return)

Returns

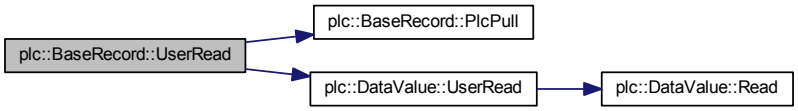
true if successfull

Definition at line 478 of file plcBase.h.

References PlcPull(), plc::DataValue::UserRead(), and value.

```
478                                                         {
479         PlcPull(); return value.UserRead (data, max); }
```

Here is the call graph for this function:



8.7.3.13 size_type plc::BaseRecord::UserReadBinary (type_binary *p*, size_type *len*) [inline]

Execute a user read, but pull plc first

Parameters

p	Pointer to data (destination buffer)
len	Length in bytes (must be the same as data length)

Returns

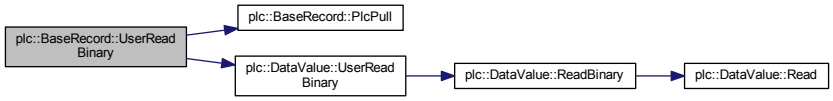
Number of bytes read (0 on error)

Definition at line 500 of file plcBase.h.

References PlcPull(), plc::DataValue::UserReadBinary(), and value.

```
500                                                     {
501         PlcPull(); return value.UserReadBinary (p, len); }
```

Here is the call graph for this function:



8.7.3.14 template<typename T > bool plc::BaseRecord::UserWrite (const T & *data*) [inline]

Execute a user write and push plc

Parameters

data	Reference to data

Returns

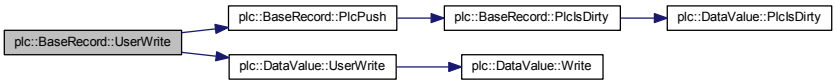
true if successfull

Definition at line 483 of file plcBase.h.

References PlcPush(), plc::DataValue::UserWrite(), and value.

```
483                                                     {
484         bool ret = value.UserWrite (data); if (ret) PlcPush(); return ret; }
```

Here is the call graph for this function:



8.7.3.15 bool plc::BaseRecord::UserWrite (const type_string_value * *data*, size_type *max*) [inline]

Execute a user write and push plc

Parameters

data	character pointer, pchar

Returns

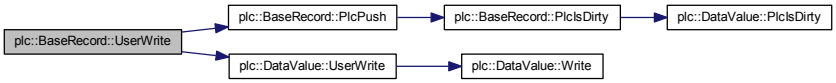
true if successfull

Definition at line 488 of file plcBase.h.

References PlcPush(), plc::DataValue::UserWrite(), and value.

```
488                                                                 {
489         bool ret = value.UserWrite (data, max); if (ret) PlcPush(); return ret; }
```

Here is the call graph for this function:



8.7.3.16 bool plc::BaseRecord::UserWrite (const type_wstring_value * *data*, size_type *max*) [inline]

Execute a user write and push plc

Parameters

data	character pointer, pwchar

Returns

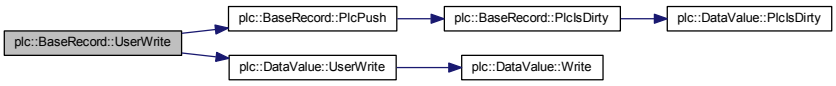
true if successfull

Definition at line 493 of file plcBase.h.

References PlcPush(), plc::DataValue::UserWrite(), and value.

```
493                                                                 {
494         bool ret = value.UserWrite (data, max); if (ret) PlcPush(); return ret; }
```

Here is the call graph for this function:



8.7.3.17 size_type plc::BaseRecord::UserWriteBinary (const type_binary *p*, size_type *len*) [inline]

Execute a user write and push plc

Parameters

p	Pointer to data (source buffer)
len	Length in bytes (must be the same as data length)

Returns

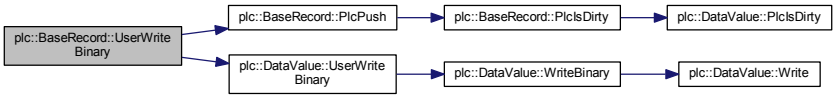
Number of bytes written (0 on error)

Definition at line 506 of file plcBase.h.

References PlcPush(), plc::DataValue::UserWriteBinary(), and value.

```
506                                                                 {
507         size_type ret = value.UserWriteBinary (p, len);
508         if (ret > 0) PlcPush(); return ret; }
```

Here is the call graph for this function:



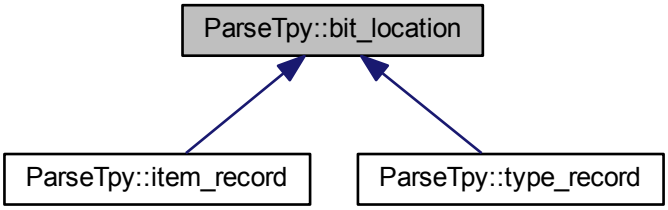
The documentation for this class was generated from the following files:

- plcBase.h
- plcBaseTemplate.h

8.8 ParseTpy::bit_location Class Reference

`#include <ParseTpy.h>`

Inheritance diagram for ParseTpy::bit_location:



Public Member Functions

- bit_location ()

  Default constructor.

- bit_location (int bo, int bs)

  Constructor.
- bool isValid () const

  Validity.
- const int get_bit_offset () const

  Get bit offset.
- void set_bit_offset (int ofs)

  Set bit offset.
- const int get_bit_size () const

  Get bit size.
- void set_bit_size (int size)

  Set bit size.

Protected Attributes

- int bitoffs

  bit offset where elements is stored
- int bitsize

  size in number of bits of symbol

8.8.1 Detailed Description

This is a base class for storing name, type, type id and opc list

Definition at line 223 of file ParseTpy.h.

The documentation for this class was generated from the following file:

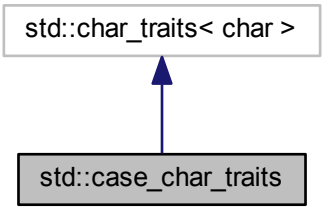
- ParseTpy.h

8.9 std::case_char_traits Struct Reference

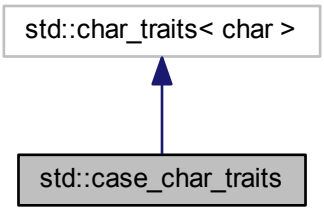
case insensitive traits.

`#include <stringcase.h>`

Inheritance diagram for std::case_char_traits:



Collaboration diagram for std::case_char_traits:



Static Public Member Functions

- static bool **eq** (const char_type &c1, const char_type &c2)
- static bool **ne** (const char_type &c1, const char_type &c2)
- static bool **lt** (const char_type &c1, const char_type &c2)
- static int **compare** (const char_type ∗s1, const char_type ∗s2, size_t n)

8.9.1 Detailed Description

case insensitive traits.

This traits class is not case sensitive.

Definition at line 38 of file stringcase.h.

The documentation for this struct was generated from the following file:

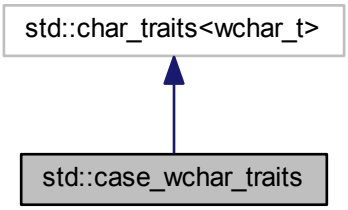
- stringcase.h

8.10 std::case_wchar_traits Struct Reference

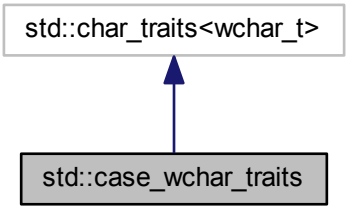
case insensitive unicode traits.

`#include <stringcase.h>`

Inheritance diagram for std::case_wchar_traits:



Collaboration diagram for std::case_wchar_traits:



Static Public Member Functions

- static bool **eq** (const char_type &c1, const char_type &c2)
- static bool **ne** (const char_type &c1, const char_type &c2)
- static bool **lt** (const char_type &c1, const char_type &c2)
- static int **compare** (const char_type *s1, const char_type *s2, size_t n)

8.10.1 Detailed Description

case insensitive unicode traits.

This unicode traits class is not case sensitive.

Definition at line 57 of file stringcase.h.

The documentation for this struct was generated from the following file:

- stringcase.h

8.11 TcComms::DataPar Struct Reference

Memory location struct.

```
#include <tcComms.h>
```

Public Attributes

- unsigned long indexGroup

  index group in ADS server
- unsigned long indexOffset

  index offset in ADS server
- unsigned long length

  count of bytes to read

8.11.1 Detailed Description

Memory location struct.

Struct for storing index group, index offset, and size of a TC symbol

Definition at line 61 of file tcComms.h.

The documentation for this struct was generated from the following file:

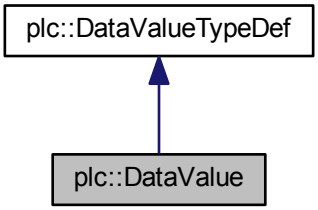
- tcComms.h

8.12 plc::DataValue Class Reference

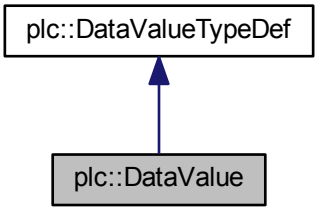
data value

```
#include <plcBase.h>
```

Inheritance diagram for plc::DataValue:



Collaboration diagram for plc::DataValue:



Public Types

- typedef void * data_type

  Internally used storage pointer type.

Public Member Functions

- DataValue ()

  Default constructor.
- DataValue (data_type_enum rt, size_type len=0)
- ~DataValue ()

  Desctructor.
- DataValue (const DataValue &)

  Copy constructor.
- DataValue & operator= (const DataValue &)

  Assignment operator.
- void Init (data_type_enum rt, size_type len=0)
- bool IsValid () const

  is valid
- data_type_enum get_data_type () const

  get type
- size_type get_size () const

  get size
- template<typename T >
  bool UserRead (T &data) const
- template<size_type N>
  bool UserRead (type_string_value(&data)[N]) const
- bool UserRead (type_string_value *data, size_type max) const
- bool UserRead (type_wstring_value *data, size_type max) const
- template<typename T >
  bool UserWrite (const T &data)
- template<size_type N>
  bool UserWrite (const type_string_value(&data)[N])
- bool UserWrite (const type_string_value *data, size_type max)
- bool UserWrite (const type_wstring_value *data, size_type max)
- size_type UserReadBinary (type_binary p, size_type len) const
- size_type UserWriteBinary (const type_binary p, size_type len)
- bool UserIsDirty () const

  New data for user.
- template<typename T >
  bool PlcRead (T &data) const
- template<size_type N>
  bool PlcRead (type_string_value(&data)[N]) const
- bool PlcRead (type_string_value *data, size_type max) const
- bool PlcRead (type_wstring_value *data, size_type max) const
- template<typename T >
  bool PlcWrite (const T &data)
- template<size_type N>
  bool PlcWrite (const type_string_value(&data)[N])
- bool PlcWrite (const type_string_value *data, size_type max)
- bool PlcWrite (const type_wstring_value *data, size_type max)
- size_type PlcReadBinary (type_binary p, size_type len) const
- size_type PlcWriteBinary (const type_binary p, size_type len)
- bool PlcIsDirty () const

  New data for plc.

Protected Member Functions

- template<typename T >
  bool Read (atomic_bool &dirty, T &data) const
- bool Read (atomic_bool &dirty, type_string &data) const
- bool Read (atomic_bool &dirty, type_wstring &data) const
- bool Read (atomic_bool &dirty, type_string_value *data, int max) const
- bool Read (atomic_bool &dirty, type_wstring_value *data, int max) const
- template<typename T >
  bool Write (atomic_bool &dirty, atomic_bool &pend, const T &data)
- bool Write (atomic_bool &dirty, atomic_bool &pend, const type_string &data)
- bool Write (atomic_bool &dirty, atomic_bool &pend, const type_wstring &data)
- bool Write (atomic_bool &dirty, atomic_bool &pend, const type_string_value *data, int max)
- bool Write (atomic_bool &dirty, atomic_bool &pend, const type_wstring_value *data, int max)
- size_type ReadBinary (atomic_bool &dirty, type_binary p, size_type len) const
- size_type WriteBinary (atomic_bool &dirty, atomic_bool &pend, const type_binary p, size_type len)

Protected Attributes

- data_type mydata

  Data pointer.
- size_type mysize
- data_type_enum mytype

  Data type.
- atomic_bool myuserdirty

  Dirty flag indicating user needs to update.
- atomic_bool myplcdirty

  Dirty flag indicating plc needs to update.

Additional Inherited Members

8.12.1 Detailed Description

data value

Class for data value This class stores a data value and provides synchronization between the user (slave) and the plc (master) interfaces. When the plc writes a value, it is marked dirty on the user side. Then, when the user reads this data, it resets the dirty flag. The same logic applies for writes by the user and reads by the plc.

Data access is guaranteed to be atomic and MT safe. Construction, initialization and destruction is not MT safe and all data access has to be stopped during these operations.

Type conversion is provided between all simple data types. However, the loss of information is not checked upon a down cast. Strings have to be read as strings. Binary data needs to be acessed with the binary read/write operations. However, all data can be accessed through binary access.

Definition at line 203 of file plcBase.h.

8.12.2 Constructor & Destructor Documentation

8.12.2.1 plc::DataValue::DataValue (data_type_enum *rt*, size_type *len* = 0) [inline],[explicit]

Constructor

Parameters

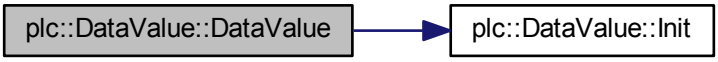
rt	Data type enumeration value

Definition at line 213 of file plcBase.h.

References Init().

```
214         : mydata (nullptr), mytype (dtInvalid), mysize (0) {
      Init(rt, len); }
```

Here is the call graph for this function:



8.12.3 Member Function Documentation

8.12.3.1 void plc::DataValue::Init (data_type_enum *rt*, size_type *len* = 0)

Initializes data value

Parameters

rt	Data type enumeration value
len	Length of data (use only for binary)

Definition at line 168 of file plcBase.cpp.

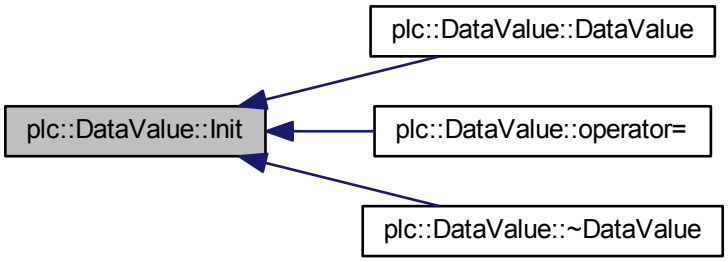
References plc::dtBinary, plc::dtBool, plc::dtDouble, plc::dtFloat, plc::dtInt16, plc::dtInt32, plc::dtInt64, plc::dtInt8, plc::dtInvalid, plc::dtString, plc::dtUInt16, plc::dtUInt32, plc::dtUInt64, plc::dtUInt8, plc::dtWString, mydata, myplcdirty, mysize, mytype, and myuserdirty.

Referenced by DataValue(), operator=(), and ~DataValue().

```
169 {
170     if (rt == mytype) {
171         if ((mytype == dtInvalid) && !mydata) return;
172         if ((mytype != dtInvalid) && mydata) return;
173     }
174     if (mydata) {
175         if (mytype == dtBinary) {
176             delete [] mydata;
177         }
178         else {
179             delete mydata;
180         }
181     }
182     mydata = nullptr;
183     mytype = rt;
184     switch (mytype)
185     {
186     case dtInvalid:
187         break;
188     case dtBool:
189         mydata = (data_type) new (std::nothrow) atomic_bool;
190         mysize = sizeof (bool);
191         break;
192     case dtInt8:
193         mydata = (data_type) new (std::nothrow) atomic_int8;
194         mysize = 1;
195         break;
196     case dtUInt8:
```

```
        mydata = (data_type) new (std::nothrow) atomic_uint8;
        mysize = 1;
        break;
    case dtInt16:
        mydata = (data_type) new (std::nothrow) atomic_int16;
        mysize = 2;
        break;
    case dtUInt16:
        mydata = (data_type) new (std::nothrow) atomic_uint16;
        mysize = 2;
        break;
    case dtInt32:
        mydata = (data_type) new (std::nothrow) atomic_int32;
        mysize = 4;
        break;
    case dtUInt32:
        mydata = (data_type) new (std::nothrow) atomic_uint32;
        mysize = 4;
        break;
    case dtInt64:
        mydata = (data_type) new (std::nothrow) atomic_int64;
        mysize = 8;
        break;
    case dtUInt64:
        mydata = (data_type) new (std::nothrow) atomic_uint64;
        mysize = 8;
        break;
    case dtFloat:
        mydata = (data_type) new (std::nothrow) atomic_float;
        mysize = 4;
        break;
    case dtDouble:
        mydata = (data_type) new (std::nothrow) atomic_double;
        mysize = 8;
        break;
    case dtString:
        mydata = (data_type) new (std::nothrow) atomic_string;
        mysize = sizeof (atomic_string);
        break;
    case dtWString:
        mydata = (data_type) new (std::nothrow) atomic_wstring;
        mysize = sizeof (atomic_wstring);
        break;
    case dtBinary:
        mydata = (data_type) new (std::nothrow) char[len];
        mysize = len;
        break;
    }
    if (!mydata) {
        mytype = dtInvalid;
        mysize = 0;
    }
    myuserdirty.store (false);
    myplcdirty.store (false);
}
```

Here is the caller graph for this function:



8.12.3.2 template<typename T > bool plc::DataValue::PlcRead (T & *data*) const [inline]

Read data by the plc

Parameters

data	Data value reference (return)

Definition at line 287 of file plcBase.h.

References myplcdirty, and Read().

Referenced by plc::BaseRecord::PlcRead().

```
287                                                     {
288         return Read (myplcdirty, data); }
```

Here is the call graph for this function:

Here is the caller graph for this function:

8.12.3.3 template<size_type N> bool plc::DataValue::PlcRead (type_string_value(&) *data[N]*) const [inline]

Read fixed length character array data by the plc

Parameters

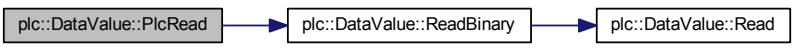
data	Data value reference for a fixed length character array (return)

Definition at line 291 of file plcBase.h.

References myplcdirty, and ReadBinary().

```
291                                                                     {
292         return ReadBinary (myplcdirty, &data, N) > 0; }
```

Here is the call graph for this function:



8.12.3.4 bool plc::DataValue::PlcRead (type_string_value * *data*, size_type *max*) const [inline]

Read character array (pchar) by the plc

Parameters

pchar	Destination buffer
max	Maximum length

Definition at line 296 of file plcBase.h.

References myplcdirty, and Read().

```
296                                                         {
297         return Read (myplcdirty, data, max); }
```

Here is the call graph for this function:

8.12.3.5 bool plc::DataValue::PlcRead (type_wstring_value * *data*, size_type *max*) const [inline]

Read character array (pwchar) by the plc

Parameters

pchar	Destination buffer
max	Maximum length

Definition at line 301 of file plcBase.h.

References myplcdirty, and Read().

```
301                                                         {
302         return Read (myplcdirty, data, max); }
```

Here is the call graph for this function:

8.12.3.6 size_type plc::DataValue::PlcReadBinary (type_binary *p*, size_type *len*) const [inline]

Read data as binary by the plc

Parameters

p	value pointer (destination buffer)
len	Length in bytes

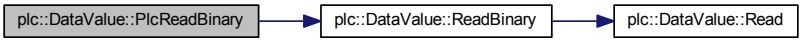
Definition at line 325 of file plcBase.h.

References myplcdirty, and ReadBinary().

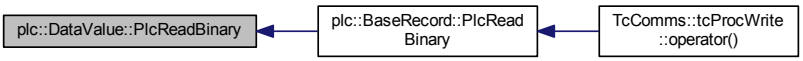
Referenced by plc::BaseRecord::PlcReadBinary().

```
325                                                                 {
326         return ReadBinary (myplcdirty, p, len); }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.12.3.7 template<typename T > bool plc::DataValue::PlcWrite (const T & *data*) [inline]

Write data by the plc

Parameters

data	Data value reference

Definition at line 305 of file plcBase.h.

References myplcdirty, myuserdirty, and Write().

Referenced by plc::BaseRecord::PlcWrite().

```
305                                                         {
306         return Write (myuserdirty, myplcdirty, data); }
```

Here is the call graph for this function:

Here is the caller graph for this function:

8.12.3.8 template<size_type N> bool plc::DataValue::PlcWrite (const type_string_value(&) *data[N]*) [inline]

Write fixed length character array data by the plc

Parameters

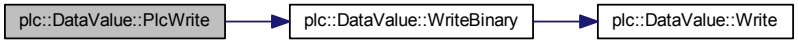
data	Data value reference

Definition at line 309 of file plcBase.h.

References myplcdirty, myuserdirty, and WriteBinary().

```
309                                                                 {
310         return WriteBinary (myuserdirty, myplcdirty, &data, N) > 0; }
```

Here is the call graph for this function:



8.12.3.9 bool plc::DataValue::PlcWrite (const type_string_value * *data,* size_type *max*) [inline]

Write character array (pchar) by the plc

Parameters

pchar	Source buffer
max	Maximum length

Definition at line 314 of file plcBase.h.

References myplcdirty, myuserdirty, and Write().

```
314                                                         {
315         return Write (myuserdirty, myplcdirty, data, max); }
```

Here is the call graph for this function:

8.12.3.10 bool plc::DataValue::PlcWrite (const type_wstring_value * *data*, size_type *max*) [inline]

Write character array (wpchar) by the plc

Parameters

pchar	Source buffer
max	Maximum length

Definition at line 319 of file plcBase.h.

References myplcdirty, myuserdirty, and Write().

```
319                                                         {
320         return Write (myuserdirty, myplcdirty, data, max); }
```

Here is the call graph for this function:

8.12.3.11 size_type plc::DataValue::PlcWriteBinary (const type_binary *p*, size_type *len*) [inline]

Write data as binary by the plc

Parameters

p	value pointer (source buffer)
len	Length in bytes

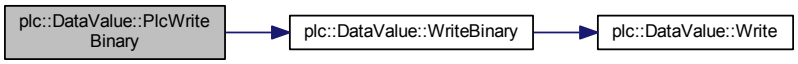
Definition at line 330 of file plcBase.h.

References myplcdirty, myuserdirty, and WriteBinary().

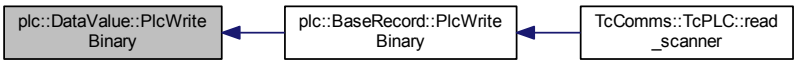
Referenced by plc::BaseRecord::PlcWriteBinary().

```
330                                                         {
331         return WriteBinary (myuserdirty, myplcdirty, p, len); }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.12.3.12 template<typename T > bool plc::DataValue::Read (atomic_bool & *dirty*, T & *data*) const [protected]

Read data

Parameters

dirty	Reference to dirty flag (user or plc)
data	Data value reference (return)

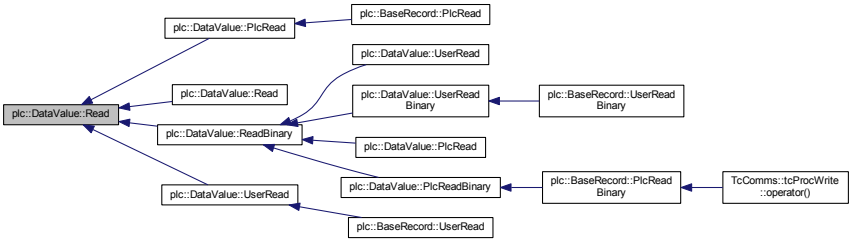
Definition at line 83 of file plcBaseTemplate.h.

References plc::dtBool, plc::dtDouble, plc::dtFloat, plc::dtInt16, plc::dtInt32, plc::dtInt64, plc::dtInt8, plc::dtUInt16, plc::dtUInt32, plc::dtUInt64, plc::dtUInt8, mydata, and mytype.

Referenced by PlcRead(), Read(), ReadBinary(), and UserRead().

```
{
    switch (mytype) {
    case dtBool:
        return reset_and_read (dirty, data, (atomic_bool*)mydata);
    case dtInt8:
        return reset_and_read (dirty, data, (atomic_int8*)mydata);
    case dtUInt8:
        return reset_and_read (dirty, data, (atomic_uint8*)mydata);
    case dtInt16:
        return reset_and_read (dirty, data, (atomic_int16*)mydata);
    case dtUInt16:
        return reset_and_read (dirty, data, (atomic_uint16*)mydata);
    case dtInt32:
        return reset_and_read (dirty, data, (atomic_int32*)mydata);
    case dtUInt32:
        return reset_and_read (dirty, data, (atomic_uint32*)mydata);
    case dtInt64:
        return reset_and_read (dirty, data, (atomic_int64*)mydata);
    case dtUInt64:
        return reset_and_read (dirty, data, (atomic_uint64*)mydata);
    case dtFloat:
        return reset_and_read (dirty, data, (atomic_float*)mydata);
    case dtDouble:
        return reset_and_read (dirty, data, (atomic_double*)mydata);
    default:
        return false;
    }
}
```

Here is the caller graph for this function:



8.12.3.13 bool plc::DataValue::Read (atomic_bool & *dirty*, type_string & *data*) const [protected]

Read string (template specialization)

Parameters

dirty	Reference to dirty flag (user or plc)
data	Data value reference (return)

Definition at line 255 of file plcBase.cpp.

References plc::dtString, mydata, and mytype.

```
{
    switch (mytype) {
    case dtString:
        return reset_and_read (dirty, data, (atomic_string*)mydata);
    default:
        return false;
    }
}
```

8.12.3.14 bool plc::DataValue::Read (atomic_bool & *dirty*, type_wstring & *data*) const [protected]

Read wstring (template specialization)

Parameters

dirty	Reference to dirty flag (user or plc)
data	Data value reference (return)

Definition at line 267 of file plcBase.cpp.

References plc::dtString, plc::dtWString, mydata, and mytype.

```
{
    switch (mytype) {
    case dtString:
        // conversion only works with simple acsii strings; not UTF-8
        return reset_and_read (dirty, data, (atomic_string*)mydata);
    case dtWString:
        return reset_and_read (dirty, data, (atomic_wstring*)
    mydata);
    default:
        return false;
    }
}
```

8.12.3.15 bool plc::DataValue::Read (atomic_bool & *dirty,* type_string_value * *data,* int *max*) const [protected]

Read character array (pchar)

Parameters

dirty	Reference to dirty flag (user or plc)
pchar	Destination buffer
max	Maximum length

Definition at line 282 of file plcBase.cpp.

References Read().

```
{
    if (!data || (max <= 0)) {
        return false;
    }
    type_string d;
    if (!Read (dirty, d)) return false;
    strncpy (data, d.c_str(), max);
    data[max-1] = 0;
    return true;
}
```

Here is the call graph for this function:

8.12.3.16 bool plc::DataValue::Read (atomic_bool & *dirty*, type_wstring_value * *data*, int *max*) const [protected]

Read character array (pwchar)

Parameters

dirty	Reference to dirty flag (user or plc)
pchar	Destination buffer
max	Maximum length

Definition at line 296 of file plcBase.cpp.

References Read().

```
{
    if (!data || (max <= 0)) {
        return false;
    }
    type_wstring d;
    if (!Read (dirty, d)) return false;
    wcsncpy (data, d.c_str(), max);
    data[max-1] = 0;
    return true;
}
```

Here is the call graph for this function:

8.12.3.17 DataValue::size_type plc::DataValue::ReadBinary (atomic_bool & *dirty*, type_binary *p*, size_type *len*) const [protected]

Read data as binary

Parameters

dirty	Reference to dirty flag (user or plc)
p	value pointer (destination buffer)
len	Length in bytes

Definition at line 364 of file plcBase.cpp.

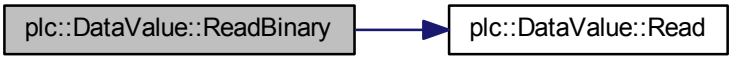
References plc::dtBinary, plc::dtBool, plc::dtDouble, plc::dtFloat, plc::dtInt16, plc::dtInt32, plc::dtInt64, plc::dtInt8, plc::dtInvalid, plc::dtString, plc::dtUInt16, plc::dtUInt32, plc::dtUInt64, plc::dtUInt8, plc::dtWString, plc::DataValueTypeDef::memory_order, mydata, mysize, mytype, and Read().

Referenced by PlcRead(), PlcReadBinary(), UserRead(), and UserReadBinary().

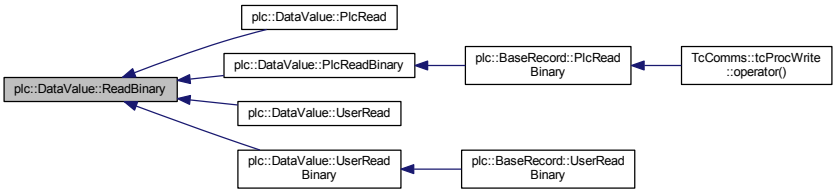
```
365 {
366     if ((mytype == dtInvalid) || !mydata || !p) {
367         return 0;
368     }
369     switch (mytype)
370     {
371     case dtBool:
372         return ((len == mysize) && Read (dirty, *(type_bool*)p)) ?
      mysize : 0;
373     case dtInt8:
374         return ((len == mysize) && Read (dirty, *(type_int8*)p)) ?
      mysize : 0;
375     case dtUInt8:
376         return ((len == mysize) && Read (dirty, *(type_uint8*)p)) ?
      mysize : 0;
377     case dtInt16:
378         return ((len == mysize) && Read (dirty, *(type_int16*)p)) ?
      mysize : 0;
379     case dtUInt16:
380         return ((len == mysize) && Read (dirty, *(type_uint16*)p)) ?
      mysize : 0;
381     case dtInt32:
382         return ((len == mysize) && Read (dirty, *(type_int32*)p)) ?
      mysize : 0;
383     case dtUInt32:
384         return ((len == mysize) && Read (dirty, *(type_uint32*)p)) ?
      mysize : 0;
385     case dtInt64:
386         return ((len == mysize) && Read (dirty, *(type_int64*)p)) ?
      mysize : 0;
387     case dtUInt64:
388         return ((len == mysize) && Read (dirty, *(type_uint64*)p)) ?
      mysize : 0;
389     case dtFloat:
390         return ((len == mysize) && Read (dirty, *(type_float*)p)) ?
      mysize : 0;
391     case dtDouble:
392         return ((len == mysize) && Read (dirty, *(type_double*)p)) ?
      mysize : 0;
393     case dtString:
394         return Read (dirty, (type_string_value*) p, len) ? len : 0;
```

```
    case dtWString:
        return Read (dirty, (type_wstring_value*) p, len / 2) ? 2 * int (len / 2)  :
    0;
    case dtBinary:
        if (len != mysize) {
            return 0;
        }
        dirty.store (false, memory_order); // must be first
        memcpy (p, (const type_binary)mydata, len);
        return mysize;
    default:
        return 0;
    }
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.12.3.18 template<typename T > bool plc::DataValue::UserRead (T & *data*) const [inline]

Read data by the user

Parameters

data	Data value reference (return)

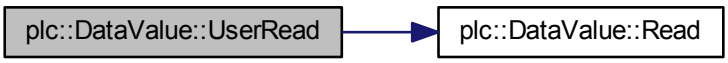
Definition at line 236 of file plcBase.h.

References myuserdirty, and Read().

Referenced by plc::BaseRecord::UserRead().

```
236                                                 {
237         return Read (myuserdirty, data); }
```

Here is the call graph for this function:



Here is the caller graph for this function:

8.12.3.19 template<size_type N> bool plc::DataValue::UserRead (type_string_value(&) *data[N]*) const [inline]

Read fixed length character array data by the user

Parameters

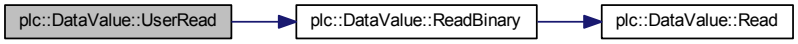
data	Data value reference for a fixed length character array (return)

Definition at line 240 of file plcBase.h.

References myuserdirty, and ReadBinary().

```
240                                                                                 {
241         return ReadBinary (myuserdirty, &data, N) > 0; }
```

Here is the call graph for this function:



8.12.3.20 bool plc::DataValue::UserRead (type_string_value * *data,* size_type *max*) const [inline]

Read character array (pchar) by the user

Parameters

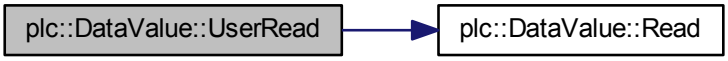
pchar	Destination buffer
max	Maximum length

Definition at line 245 of file plcBase.h.

References myuserdirty, and Read().

```
245                                                                    {
246         return Read (myuserdirty, data, max); }
```

Here is the call graph for this function:



8.12.3.21 bool plc::DataValue::UserRead (type_wstring_value * *data*, size_type *max*) const [inline]

Read character array (pwchar) by the user

Parameters

pchar	Destination buffer
max	Maximum length

Definition at line 250 of file plcBase.h.

References myuserdirty, and Read().

```
250                                                                    {
251         return Read (myuserdirty, data, max); }
```

Here is the call graph for this function:

8.12.3.22 size_type plc::DataValue::UserReadBinary (type_binary *p*, size_type *len*) const [inline]

Read data as binary by the user

Parameters

p	value pointer (destination buffer)
len	Length in bytes

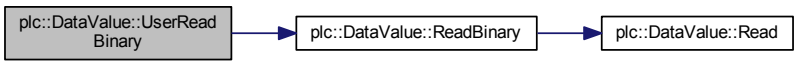
Definition at line 275 of file plcBase.h.

References myuserdirty, and ReadBinary().

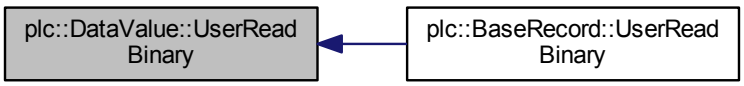
Referenced by plc::BaseRecord::UserReadBinary().

```
275                                                                    {
276         return ReadBinary (myuserdirty, p, len); }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.12.3.23 template<typename T > bool plc::DataValue::UserWrite (const T & *data*) [inline]

Write data by the user

Parameters

data	Data value reference

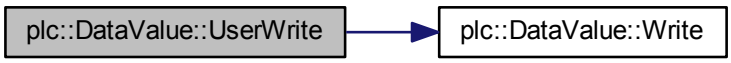
Definition at line 254 of file plcBase.h.

References myplcdirty, myuserdirty, and Write().

Referenced by plc::BaseRecord::UserWrite().

```
254                                                    {
255         return Write (myplcdirty, myuserdirty, data); }
```

Here is the call graph for this function:



Here is the caller graph for this function:

8.12.3.24 template<size_type N> bool plc::DataValue::UserWrite (const type_string_value(&) *data[N]*) [inline]

Write fixed length character array data by the user

Parameters

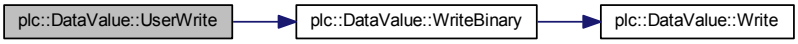
data	Data value reference

Definition at line 258 of file plcBase.h.

References myplcdirty, myuserdirty, and WriteBinary().

```
258                                                                    {
259         return WriteBinary (myplcdirty, myuserdirty, &data, N) > 0; }
```

Here is the call graph for this function:



8.12.3.25 bool plc::DataValue::UserWrite (const type_string_value * *data,* size_type *max*) [inline]

Write character array (pchar) by the user

Parameters

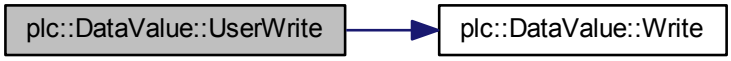
pchar	Source buffer
max	Maximum length

Definition at line 264 of file plcBase.h.

References myplcdirty, myuserdirty, and Write().

```
264                                                     {
265         return Write (myplcdirty, myuserdirty, data, max); }
```

Here is the call graph for this function:



8.12.3.26 bool plc::DataValue::UserWrite (const type_wstring_value * *data*, size_type *max*) [inline]

Write character array (wpchar) by the user

Parameters

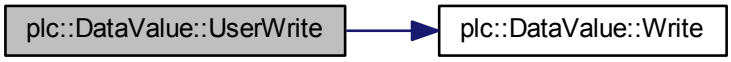
pchar	Source buffer
max	Maximum length

Definition at line 269 of file plcBase.h.

References myplcdirty, myuserdirty, and Write().

```
269                                                     {
270         return Write (myplcdirty, myuserdirty, data, max); }
```

Here is the call graph for this function:



8.12.3.27 size_type plc::DataValue::UserWriteBinary (const type_binary *p*, size_type *len*) [inline]

Write data as binary by the user

Parameters

p	value pointer (source buffer)
len	Length in bytes

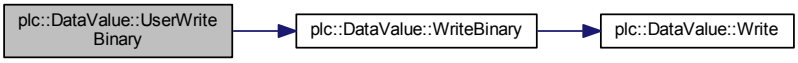
Definition at line 280 of file plcBase.h.

References myplcdirty, myuserdirty, and WriteBinary().

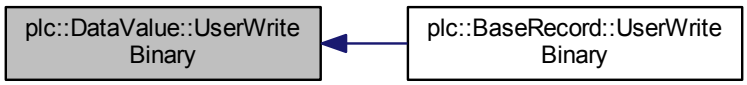
Referenced by plc::BaseRecord::UserWriteBinary().

```
280                                                     {
281         return WriteBinary (myplcdirty, myuserdirty, p, len); }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.12.3.28 template<typename T > bool plc::DataValue::Write (atomic_bool & *dirty*, atomic_bool & *pend*, const T & *data*) [protected]

Write data

Parameters

dirty	Reference to dirty flag (user or plc)
pend	Reference to pending read flag (plc or user)
data	Data value reference (return)

Definition at line 116 of file plcBaseTemplate.h.

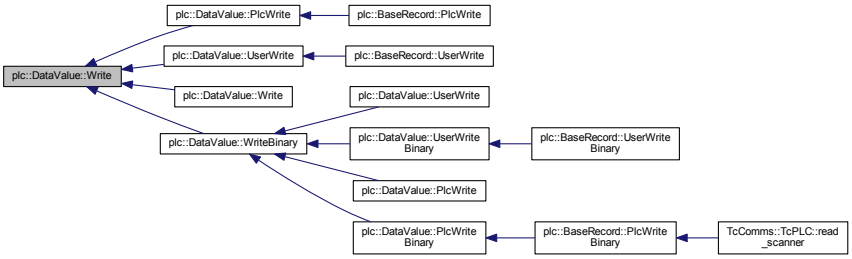
References plc::dtBool, plc::dtDouble, plc::dtFloat, plc::dtInt16, plc::dtInt32, plc::dtInt64, plc::dtInt8, plc::dtUInt16, plc::dtUInt32, plc::dtUInt64, plc::dtUInt8, mydata, and mytype.

Referenced by PlcWrite(), UserWrite(), Write(), and WriteBinary().

```
117 {
118     switch (mytype) {
119     case dtBool:
120         return write_and_test (dirty, pend, (atomic_bool*)mydata, data);
121     case dtInt8:
122         return write_and_test (dirty, pend, (atomic_int8*)mydata, data);
123     case dtUInt8:
124         return write_and_test (dirty, pend, (atomic_uint8*)mydata, data);
125     case dtInt16:
```

```
        return write_and_test (dirty, pend, (atomic_int16*)mydata, data);
    case dtUInt16:
        return write_and_test (dirty, pend, (atomic_uint16*)mydata, data);
    case dtInt32:
        return write_and_test (dirty, pend, (atomic_int32*)mydata, data);
    case dtUInt32:
        return write_and_test (dirty, pend, (atomic_uint32*)mydata, data);
    case dtInt64:
        return write_and_test (dirty, pend, (atomic_int64*)mydata, data);
    case dtUInt64:
        return write_and_test (dirty, pend, (atomic_uint64*)mydata, data);
    case dtFloat:
        return write_and_test (dirty, pend, (atomic_float*)mydata, data);
    case dtDouble:
        return write_and_test (dirty, pend, (atomic_double*)mydata, data);
    default:
        return false;
    }
}
```

Here is the caller graph for this function:



8.12.3.29 bool plc::DataValue::Write (atomic_bool & *dirty*, atomic_bool & *pend*, const type_string & *data*) [protected]

Write string (template specialization)

Parameters

dirty	Reference to dirty flag (user or plc)
pend	Reference to pending read flag (plc or user)
data	Data value reference

Definition at line 310 of file plcBase.cpp.

References plc::dtString, plc::dtWString, mydata, and mytype.

```
{
    switch (mytype) {
    case dtString:
        return write_and_test (dirty, pend, (atomic_string*)mydata, data);
    case dtWString:
        // conversion only works with simple acsii strings; not UTF-8
        return write_and_test (dirty, pend, (atomic_wstring*)
    mydata, data);
    default:
        return false;
    }
}
```

8.12.3.30 bool plc::DataValue::Write (atomic_bool & *dirty*, atomic_bool & *pend*, const type_wstring & *data*) [protected]

Write wstring (template specialization)

Parameters

dirty	Reference to dirty flag (user or plc)
pend	Reference to pending read flag (plc or user)
data	Data value reference

Definition at line 326 of file plcBase.cpp.

References plc::dtWString, mydata, and mytype.

```
{
    switch (mytype) {
    case dtWString:
        return write_and_test (dirty, pend, (atomic_wstring*)
  mydata, data);
    default:
        return false;
    }
}
```

8.12.3.31 bool plc::DataValue::Write (atomic_bool & *dirty*, atomic_bool & *pend*, const type_string_value * *data*, int *max*) [protected]

Write character array (pchar)

Parameters

dirty	Reference to dirty flag (user or plc)
pend	Reference to pending read flag (plc or user)
pchar	Source buffer
max	Maximum length

Definition at line 339 of file plcBase.cpp.

References Write().

```
{
    if (!data || (max <= 0)) {
        return false;
    }
    type_string d (data, max);
    return Write (dirty, pend, d);
}
```

Here is the call graph for this function:

8.12.3.32 bool plc::DataValue::Write (atomic_bool & *dirty*, atomic_bool & *pend*, const type_wstring_value * *data*, int *max*) [protected]

Write character array (pwchar)

Parameters

dirty	Reference to dirty flag (user or plc)
pend	Reference to pending read flag (plc or user)
pchar	Source buffer
max	Maximum length

Definition at line 351 of file plcBase.cpp.

References Write().

```
353 {
354     if (!data || (max <= 0)) {
355         return false;
356     }
357     type_wstring d (data, max);
358     return Write (dirty, pend, d);
359 }
```

Here is the call graph for this function:

8.12.3.33 DataValue::size_type plc::DataValue::WriteBinary (atomic_bool & *dirty*, atomic_bool & *pend*, const type_binary *p*, size_type *len*) [protected]

Write data as binary

Parameters

dirty	Reference to dirty flag (user or plc)
pend	Reference to pending read flag (plc or user)
p	value pointer (source buffer)
len	Length in bytes

Definition at line 412 of file plcBase.cpp.

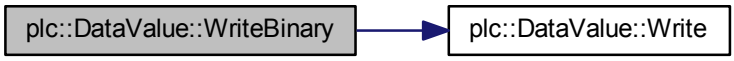
References plc::dtBinary, plc::dtBool, plc::dtDouble, plc::dtFloat, plc::dtInt16, plc::dtInt32, plc::dtInt64, plc::dtInt8, plc::dtInvalid, plc::dtString, plc::dtUInt16, plc::dtUInt32, plc::dtUInt64, plc::dtUInt8, plc::dtWString, plc::DataValueTypeDef::memory_order, mydata, mysize, mytype, and Write().

Referenced by PlcWrite(), PlcWriteBinary(), UserWrite(), and UserWriteBinary().

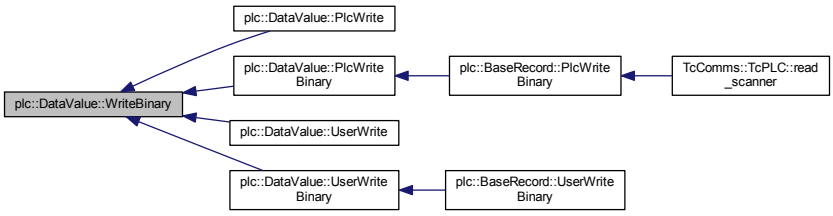
```
414 {
415     if ((mytype == dtInvalid) || !mydata || !p) {
416         return 0;
417     }
418     switch (mytype)
419     {
420     case dtBool:
421         return ((len == mysize) && Write (dirty, pend, *(const
      type_bool*)p)) ? mysize : 0;
422     case dtInt8:
423         return ((len == mysize) && Write (dirty, pend, *(const
      type_int8*)p)) ? mysize : 0;
424     case dtUInt8:
425         return ((len == mysize) && Write (dirty, pend, *(const
      type_uint8*)p)) ? mysize : 0;
```

```
    case dtInt16:
        return ((len == mysize) && Write (dirty, pend, *(const type_int16*)p)) ? mysize : 0;
    case dtUInt16:
        return ((len == mysize) && Write (dirty, pend, *(const type_uint16*)p)) ? mysize : 0;
    case dtInt32:
        return ((len == mysize) && Write (dirty, pend, *(const type_int32*)p)) ? mysize : 0;
    case dtUInt32:
        return ((len == mysize) && Write (dirty, pend, *(const type_uint32*)p)) ? mysize : 0;
    case dtInt64:
        return ((len == mysize) && Write (dirty, pend, *(const type_int64*)p)) ? mysize : 0;
    case dtUInt64:
        return ((len == mysize) && Write (dirty, pend, *(const type_uint64*)p)) ? mysize : 0;
    case dtFloat:
        return ((len == mysize) && Write (dirty, pend, *(const type_float*)p)) ? mysize : 0;
    case dtDouble:
        return ((len == mysize) && Write (dirty, pend, *(const type_double*)p)) ? mysize : 0;
    case dtString:
        return Write (dirty, pend, (const type_string_value*) p, len) ? len : 0;
    case dtWString:
        return Write (dirty, pend, (const type_wstring_value*) p, len / 2) ? 2 * int (len / 2)  : 0;
    case dtBinary:
        if (len != mysize) {
            return 0;
        }
        memcpy ((type_binary)mydata, p, len);
        dirty.store (false, memory_order); // must be last
        return mysize;
    default:
        return 0;
    }
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.12.4 Member Data Documentation

8.12.4.1 size_type plc::DataValue::mysize [protected]

Size of allocated memory for simple types; size of string class for strings and size of data for binary

Definition at line 408 of file plcBase.h.

Referenced by get_size(), Init(), IsValid(), operator=(), ReadBinary(), and WriteBinary().

The documentation for this class was generated from the following files:

- plcBase.h
- plcBase.cpp
- plcBaseTemplate.h

8.13 plc::DataValueTraits< T > Struct Template Reference

data value traits

```
#include <plcBase.h>
```

Public Types

- typedef size_t size_type

  size type
- typedef plc::data_type_enum data_type_enum

  enumerated type for data type
- typedef T traits_type

  traits type
- typedef std::atomic< T > traits_atomic

  atomic variable type

Public Member Functions

- template<>
  const DataValueTraits
  < DataValueTypeDef::type_bool >
  ::data_type_enum **data_enum**
- template<>
  const DataValueTraits
  < DataValueTypeDef::type_int8 >
  ::data_type_enum **data_enum**
- template<>
  const DataValueTraits
  < DataValueTypeDef::type_uint8 >
  ::data_type_enum **data_enum**
- template<>
  const DataValueTraits
  < DataValueTypeDef::type_int16 >
  ::data_type_enum **data_enum**
- template<>
  const DataValueTraits
  < DataValueTypeDef::type_uint16 >
  ::data_type_enum **data_enum**

- template<>
  const DataValueTraits
  < DataValueTypeDef::type_int32 >
  ::data_type_enum **data_enum**
- template<>
  const DataValueTraits
  < DataValueTypeDef::type_uint32 >
  ::data_type_enum **data_enum**
- template<>
  const DataValueTraits
  < DataValueTypeDef::type_float >
  ::data_type_enum **data_enum**
- template<>
  const DataValueTraits
  < DataValueTypeDef::type_double >
  ::data_type_enum **data_enum**
- template<>
  const DataValueTraits
  < DataValueTypeDef::type_string >
  ::data_type_enum **data_enum**
- template<>
  const DataValueTraits
  < DataValueTypeDef::type_wstring >
  ::data_type_enum **data_enum**
- template<>
  const DataValueTraits
  < DataValueTypeDef::type_binary >
  ::data_type_enum **data_enum**

Static Public Attributes

- static const data_type_enum data_enum = dtInvalid

  data type enumertaion value

8.13.1 Detailed Description

template<typename T>struct plc::DataValueTraits< T >

data value traits

Traits class for data value

Definition at line 95 of file plcBase.h.

The documentation for this struct was generated from the following files:

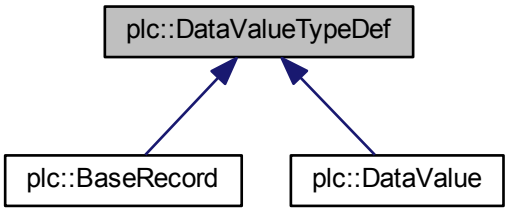
- plcBase.h
- plcBaseTemplate.h

8.14 plc::DataValueTypeDef Struct Reference

collection of type definitions

```
#include <plcBase.h>
```

Inheritance diagram for plc::DataValueTypeDef:



Public Types

- typedef size_t size_type

  size type
- typedef plc::data_type_enum data_type_enum

  enumerated type for data type
- typedef bool type_bool

  bool type
- typedef signed char type_int8

  1-byte integer type
- typedef unsigned char type_uint8

  1-byte unsigned integer type
- typedef short type_int16

  3-byte integer type
- typedef unsigned short type_uint16

  3-byte unsigned integer type
- typedef int type_int32

  4-byte integer type
- typedef unsigned int type_uint32

  4-byte unsigned integer type
- typedef long long type_int64

  8-byte integer type
- typedef unsigned long long type_uint64

  8-byte unsigned integer type
- typedef float type_float

  4-byte single precision floating point type
- typedef double type_double

  4-byte double precision floating point type
- typedef std::string type_string

  string type
- typedef std::wstring type_wstring

  wstring type
- typedef void * type_binary

  binary type
- typedef std::string::value_type type_string_value

string character type

- typedef std::wstring::value_type type_wstring_value

  wstring character type

- typedef DataValueTraits < type_bool >::traits_atomic atomic_bool

  atomic bool type

- typedef DataValueTraits < type_int8 >::traits_atomic atomic_int8

  atomic 1-byte integer type

- typedef DataValueTraits < type_uint8 >::traits_atomic atomic_uint8

  atomic 1-byte unsigned integer type

- typedef DataValueTraits < type_int16 >::traits_atomic atomic_int16

  atomic 2-byte integer type

- typedef DataValueTraits < type_uint16 >::traits_atomic atomic_uint16

  atomic 2-byte unsigned integer type

- typedef DataValueTraits < type_int32 >::traits_atomic atomic_int32

  atomic 4-byte integer type

- typedef DataValueTraits < type_uint32 >::traits_atomic atomic_uint32

  atomic 4-byte unsigned integer type

- typedef DataValueTraits < type_int64 >::traits_atomic atomic_int64

  atomic 8-byte integer type

- typedef DataValueTraits < type_uint64 >::traits_atomic atomic_uint64

  atomic 8-byte unsigned integer type

- typedef DataValueTraits < type_float >::traits_atomic atomic_float

  atomic 4-byte single precision floating point type

- typedef DataValueTraits < type_double >::traits_atomic atomic_double

  atomic 8-byte double precision floating point type

- typedef DataValueTraits < type_string >::traits_atomic atomic_string

  atomic string type

- typedef DataValueTraits < type_wstring > ::traits_atomic atomic_wstring

  atomic wstring type

- typedef DataValueTraits < type_binary >::traits_atomic atomic_binary

  atomic binary type

Static Public Attributes

- static const std::memory_order memory_order = std::memory_order_seq_cst

  memory order used for atomic access

8.14.1 Detailed Description

collection of type definitions

Type definitions for data value

Definition at line 112 of file plcBase.h.

The documentation for this struct was generated from the following file:

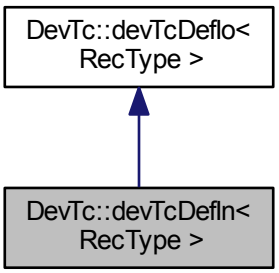
- plcBase.h

8.15 DevTc::devTcDefIn< RecType > Struct Template Reference

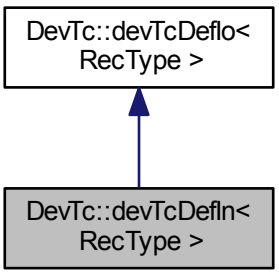
device support input record.

```
#include <devTc.h>
```

Inheritance diagram for DevTc::devTcDefIn< RecType >:



Collaboration diagram for DevTc::devTcDefIn< RecType >:



Public Member Functions

- devTcDefIn ()

Constructor.

- template<>
  long **read** (rec_type_ptr precord)
- template<>
  long **read** (rec_type_ptr precord)

Static Public Member Functions

- static long init_read_record (rec_type_ptr prec)

  init callback for read records
- static long read (rec_type_ptr precord)

  read callback

Additional Inherited Members

8.15.1 Detailed Description

template<epics_record_enum RecType>struct DevTc::devTcDefIn< RecType >

device support input record.

Deviced Support Record for TwinCAT/ADS input This structure defines the callback functions for the TC device support. This is a base class for both read and write records.

Definition at line 237 of file devTc.h.

The documentation for this struct was generated from the following files:

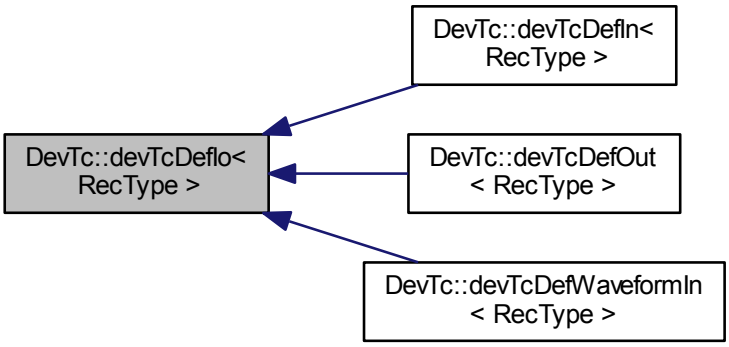
- devTc.h
- devTcTemplate.h

8.16 DevTc::devTcDefIo< RecType > Struct Template Reference

device support record.

```
#include <devTc.h>
```

Inheritance diagram for DevTc::devTcDefIo< RecType >:



Public Types

- typedef epics_record_traits < RecType >::traits_type rec_type

  Record type: aiRecord, etc.
- typedef rec_type * rec_type_ptr

  Pointer to record type.

Public Attributes

- long number

  Number of support functions.
- DEVSUPFUN report_fn

  Report support function.
- DEVSUPFUN init_fn

  Init support function.
- DEVSUPFUN init_record_fn

  Record init support function.
- DEVSUPFUN get_ioint_info_fn

  IO/INT support function.
- DEVSUPFUN io_fn

  Read/write support function.
- DEVSUPFUN special_linconv_fn

  Linear conversion support function.

Protected Member Functions

- devTcDefIo ()

  Hide constructor.

Static Protected Member Functions

- static long init (int after)

  init callback
- static long get_ioint_info (int cmd, dbCommon *prec, IOSCANPVT *ppvt)

  IO/INT info callback.

Static Protected Attributes

- static int initCounter = 0

  init counter

8.16.1 Detailed Description

template<epics_record_enum RecType>struct DevTc::devTcDefIo< RecType >

device support record.

Deviced Support Record for generic TwinCAT/ADS IO This structure defines the callback functions for the TC device support. This is a base class for both read and write records.

Definition at line 197 of file devTc.h.

The documentation for this struct was generated from the following files:

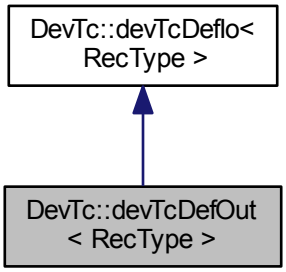
- devTc.h
- devTcTemplate.h

8.17 DevTc::devTcDefOut< RecType > Struct Template Reference

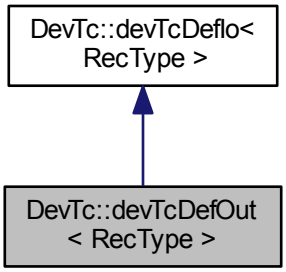
device support output record.

`#include <devTc.h>`

Inheritance diagram for DevTc::devTcDefOut< RecType >:



Collaboration diagram for DevTc::devTcDefOut< RecType >:



Public Member Functions

- devTcDefOut ()

  Constructor.
- template<>
  long **write** (rec_type_ptr precord)
- template<>
  long **write** (rec_type_ptr precord)

Static Public Member Functions

- static long init_write_record (rec_type_ptr prec)

  init callback for write records

- static long write (rec_type_ptr precord)

  write callback

Additional Inherited Members

8.17.1 Detailed Description

template<epics_record_enum RecType>struct DevTc::devTcDefOut< RecType >

device support output record.

Deviced Support Record for TwinCAT/ADS output This structure defines the callback functions for the TC device support. This is a base class for both read and write records.

Definition at line 253 of file devTc.h.

The documentation for this struct was generated from the following files:

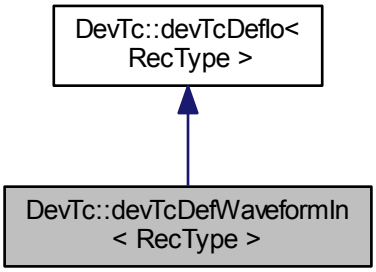
- devTc.h
- devTcTemplate.h

8.18 DevTc::devTcDefWaveformIn< RecType > Struct Template Reference

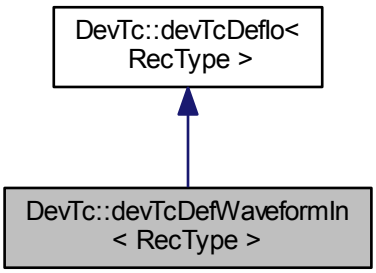
device support waveform record.

`#include <devTc.h>`

Inheritance diagram for DevTc::devTcDefWaveformIn< RecType >:



Collaboration diagram for DevTc::devTcDefWaveformIn< RecType >:



Public Member Functions

- devTcDefWaveformIn ()

  Constructor.

Static Public Member Functions

- static long init_read_waveform_record (rec_type_ptr prec)

  init callback for read records
- static long read_waveform (rec_type_ptr precord)

  read callback

Additional Inherited Members

8.18.1 Detailed Description

template<epics_record_enum RecType>struct DevTc::devTcDefWaveformIn< RecType >

device support waveform record.

Deviced Support Record for TwinCAT/ADS waveform input This structure defines the callback functions for the TC device support. This is a base class for both read and write records.

Definition at line 269 of file devTc.h.

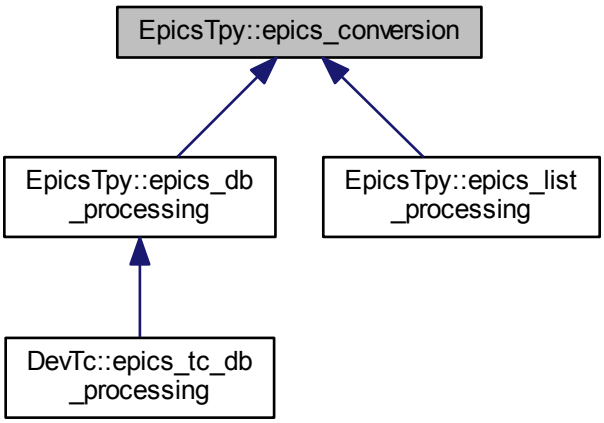
The documentation for this struct was generated from the following files:

- devTc.h
- devTcTemplate.h

8.19 EpicsTpy::epics_conversion Class Reference

```
#include <TpyToEpics.h>
```

Inheritance diagram for EpicsTpy::epics_conversion:



Public Member Functions

- epics_conversion ()

  Default constructor.
- epics_conversion (case_type caseconv, bool noindex)
- epics_conversion (tc_epics_conv epics_conv, case_type caseconv, bool noldot, bool noindex)
- tc_epics_conv get_conversion_rule () const

  Get the conversion rule.
- void set_conversion_rule (tc_epics_conv epics_conv)

  Set the conversion rule.
- case_type get_case_rule () const

  Get the conversion rule.
- void set_case_rule (case_type epics_conv)

  Set the conversion rule.
- bool get_dot_rule () const

  Get the leadin dot rule.
- void set_dot_rule (bool noldot)

  Set the leading dot rule.
- bool get_array_rule () const

  Get the array index rule.
- void set_array_rule (bool noindex)

  Set the array conversion rule.
- std::string to_epics (const std::stringcase &name) const

Protected Attributes

- tc_epics_conv conv_rule

  Conversion rule.
- case_type case_epics_names

Case conversion rule.

- bool no_leading_dot

  Leading dot conversion rule.

- bool no_array_index

  Array index conversion rule.

8.19.1 Detailed Description

Epics channel conversion arguments Epics channels are generated from opc through a conversion rule

Definition at line 50 of file TpyToEpics.h.

8.19.2 Constructor & Destructor Documentation

8.19.2.1 EpicsTpy::epics_conversion::epics_conversion (case_type *caseconv*, bool *noindex*) [inline]

Constructor

Parameters

caseconv	Case conversion specification
noindex	Eliminate array indices '[n]' with '_n'

Definition at line 59 of file TpyToEpics.h.

```
60            : conv_rule (ligo_std), case_epics_names (caseconv),
61            no_leading_dot (true), no_array_index (noindex) {}
```

8.19.2.2 EpicsTpy::epics_conversion::epics_conversion (tc_epics_conv *epics_conv*, case_type *caseconv*, bool *noldot*, bool *noindex*) [inline]

Constructor

Parameters

epics_conv	Epics conversion rule
caseconv	Case conversion specification
noldot	Eliminate leading dot in a name
noindex	Eliminate array indices '[n]' with '_n'

Definition at line 67 of file TpyToEpics.h.

```
69            : conv_rule (epics_conv), case_epics_names (caseconv),
70            no_leading_dot (noldot), no_array_index (noindex) {}
```

8.19.3 Member Function Documentation

8.19.3.1 string EpicsTpy::epics_conversion::to_epics (const std::stringcase & *name*) const

Converts a TwinCAT or OPC name to an EPICS channel name

Parameters

name	TwinCAT/opc name

Returns

EPICS name

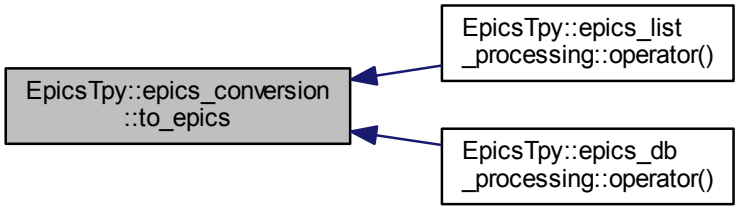
Definition at line 20 of file TpyToEpics.cpp.

References EpicsTpy::ligo_std, EpicsTpy::no_conversion, EpicsTpy::no_dot, EpicsTpy::preserve_case, and EpicsTpy::upper_case.

Referenced by EpicsTpy::epics_list_processing::operator()(), and EpicsTpy::epics_db_processing::operator()().

```
21     {
22         stringcase n (name);
23         stringcase::size_type pos;
24
25         // eliminate leading dot
26         if (no_leading_dot || (conv_rule == ligo_std)) {
27             if (!n.empty() && (n[0] == '.')) {
28                 n.erase (0, 1);
29             }
30         }
31
32         // apply conversion rules
33         switch (conv_rule) {
34         // ligo standard
35         case ligo_std:
36             // replace first dot with colon
37             pos = n.find ('.');
38             if (pos != stringcase::npos) {
39                 n[pos] = ':';
40             }
41             // replace second dot with dash
42             pos = n.find ('.');
43             if (pos != stringcase::npos) {
44                 n[pos] = '-';
45             }
46             // replace remaining dots with underscore
47             while ((pos = n.find ('.')) != stringcase::npos) {
48                 n[pos] = '_';
49             }
50             break;
51         // replace all dots with underscores
52         case no_dot:
53             while ((pos = n.find ('.')) != stringcase::npos) {
54                 n[pos] = '_';
55             }
56             break;
57         // do nothing
58         case no_conversion:
59         default:
60             break;
61         }
62         // force case if necessary
63         if (case_epics_names != preserve_case) {
64             for (pos = 0; pos < n.size(); ++pos) {
65                 n[pos] = (case_epics_names == upper_case) ? toupper (n[pos]) :
    tolower (n[pos]);
66             }
67         }
68         // replace array brackets with underscore if necessary
69         if (no_array_index) {
70             while ((pos = n.find ('[')) != stringcase::npos) {
71                 n[pos] = '_';
72             }
73             while ((pos = n.find (']')) != stringcase::npos) {
74                 n.erase (pos, 1);
75             }
76         }
77         return string (n.c_str());
78     }
```

Here is the caller graph for this function:



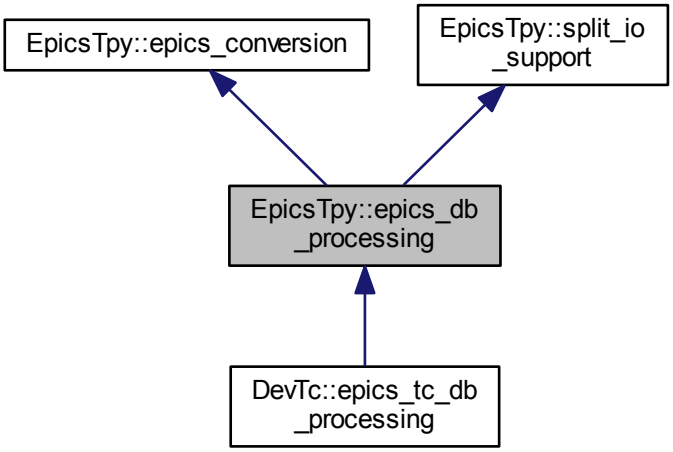
The documentation for this class was generated from the following files:

- TpyToEpics.h
- TpyToEpics.cpp

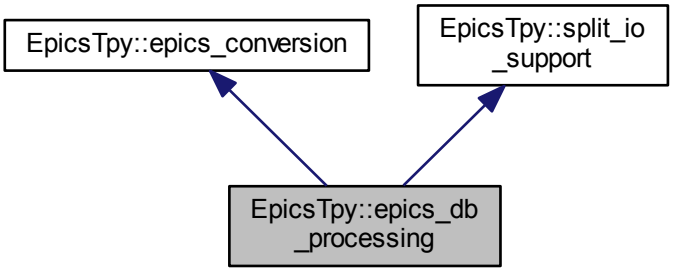
8.20 EpicsTpy::epics_db_processing Class Reference

```
#include <TpyToEpics.h>
```

Inheritance diagram for EpicsTpy::epics_db_processing:



Collaboration diagram for EpicsTpy::epics_db_processing:



Public Member Functions

- epics_db_processing ()

  Default constructor.
- device_support_type get_device_support () const

  Get device support conversion rule.
- void set_device_support (device_support_type devsup)

  Set device support conversion rule.
- bool operator() (const ParseTpy::process_arg &arg)

Protected Member Functions

- bool process_field_string (std::stringcase name, std::stringcase val)
- bool process_field_numeric (std::stringcase name, int val)
- bool process_field_numeric (std::stringcase name, std::stringcase val)
- bool process_field_alarm (std::stringcase name, std::stringcase severity)

Protected Attributes

- device_support_type device_support

  Device support field conversion rule.

8.20.1 Detailed Description

Class for generatig an EPICS database record

Definition at line 274 of file TpyToEpics.h.

8.20.2 Member Function Documentation

8.20.2.1 bool EpicsTpy::epics_db_processing::operator() (const ParseTpy::process_arg & *arg*)

Process a variable

Parameters

arg	Process argument describign the variable and type

Returns

True if successfully processed

Definition at line 331 of file TpyToEpics.cpp.

References device_support, EpicsTpy::device_support_opc_name, EpicsTpy::device_support_tc_name, EpicsTpy::split_io_support::get_file(), ParseTpy::process_arg::get_full(), ParseTpy::base_record::get_name(), ParseTpy::base_record::get_opc(), ParseTpy::process_arg::get_process_type(), ParseTpy::opc_list::get_properties(), ParseTpy::opc_list::get_property(), ParseTpy::base_record::get_type_name(), EpicsTpy::split_io_support::increment(), ParseTpy::process_arg::is_atomic(), ParseTpy::opc_list::is_readonly(), process_field_alarm(), process_field_numeric(), process_field_string(), ParseTpy::pt_bool, ParseTpy::pt_enum, ParseTpy::pt_int, ParseTpy::pt_real, ParseTpy::pt_string, and EpicsTpy::epics_conversion::to_epics().

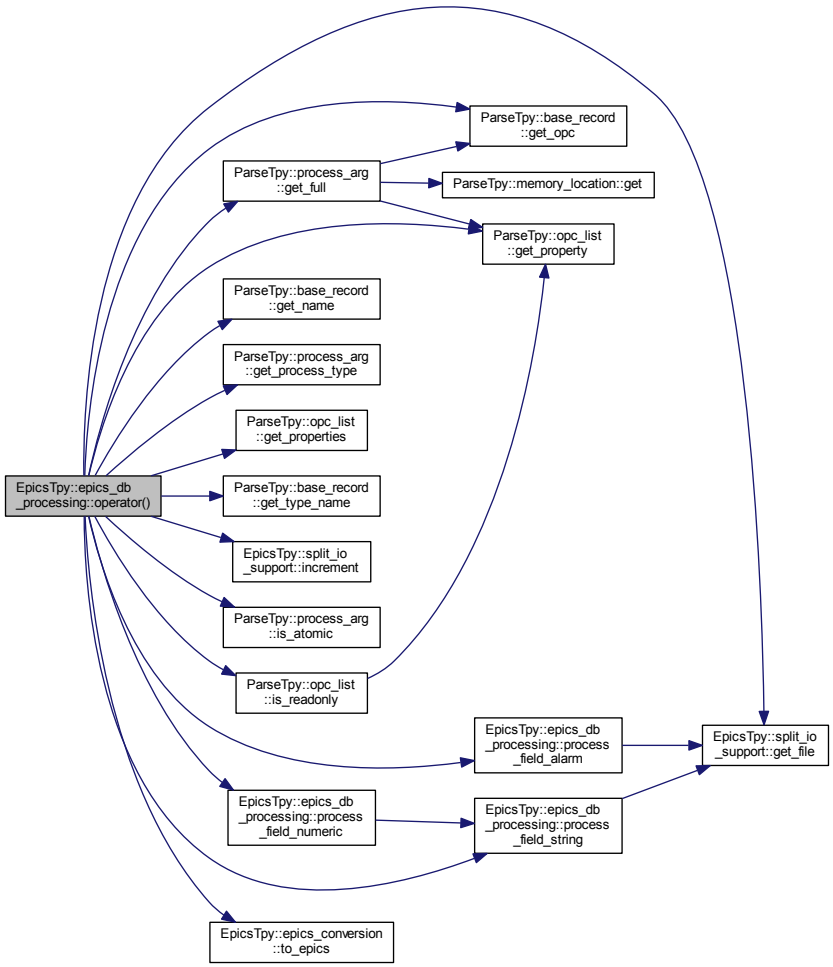
```
{
    // quit if not atomic
    if (!arg.is_atomic()) {
        return false;
    }

    // readonly?
    bool readonly = arg.get_opc().is_readonly();
    increment (readonly);

    // default process type conversion
    stringcase tname;
    switch (arg.get_process_type()) {
    case pt_int:
        tname = readonly ? "longin" : "longout";
        break;
    case pt_real:
        tname = readonly ? "ai" : "ao";
        break;
    case pt_bool:
        tname = readonly ? "bi" : "bo";
        break;
    case pt_string:
        tname = readonly ? "stringin" : "stringout";
        break;
    case pt_enum:
        tname = readonly ? "mbbi" : "mbbo";
        break;
    default:
        fprintf (stderr, "Unknown type %s for %s\n",
            arg.get_type_name().c_str(), arg.get_name().c_str());
        return false;
    }
    // check OPC_PROP_RECTYPE
    stringcase s;
    if (arg.get_opc().get_property (OPC_PROP_RECTYPE, s)) {
        tname = s;
    }

    // check record header
    string epicsname = to_epics (arg.get_name());
    if (epicsname.size() > MAX_EPICS_CHANNEL) {
        fprintf (stderr, "Warning: channel name %s too long by %i\n",
            epicsname.c_str(), epicsname.size() - MAX_EPICS_CHANNEL);
        return false;
    }
    // now print header
    fprintf(get_file(), "record(%s,\"%s\") {\n", tname.c_str(), epicsname.c_str());

    // check OPC_PROP_DESC
    if (arg.get_opc().get_property (OPC_PROP_DESC, s)) {
        if (s.size() > MAX_EPICS_DESC) {
            fprintf (stderr, "Warning: DESC for %s too long by %i\n",
                arg.get_name().c_str(), s.size() - MAX_EPICS_DESC);
        }
        process_field_string (EPICS_DB_DESC, s);
    }
    // add SCAN
    process_field_string (EPICS_DB_SCAN, readonly ? "I/O Intr" : "Passive");
    // check for DTYP
    stringcase dtyp = device_support ==
device_support_tc_name ? "tcat" : "opc";
```

```
    if (arg.get_opc().get_property (OPC_PROP_DTYP, s)) {
        if (s.find ("raw") == stringcase::npos) {
            dtyp = device_support == device_support_tc_name ? "tcat raw
" : "opcRaw";
        }
    }
    process_field_string (EPICS_DB_DTYP, dtyp);
    // INPUT/OUTPUT field
    stringcase inplink;
    stringcase servername;
    switch (device_support) {
    case device_support_opc_name:
    default:
        // check for server
        servername = "opc";
        arg.get_opc().get_property (OPC_PROP_SERVER, servername);
        // input/output link
        inplink = stringcase ("@") + servername + arg.get_name();
        break;
    case device_support_tc_name:
        // input/output link
        inplink = stringcase ("@") + arg.get_full();
        break;
    }
    process_field_string (readonly ? EPICS_DB_INP : EPICS_DB_OUT, inplink);

    // check for TSE
    int tse = -2;
    arg.get_opc().get_property (OPC_PROP_TSE, tse);
    process_field_numeric (EPICS_DB_TSE, tse);
    // check for PINI
    int pini = readonly ? 1 : 0;
    arg.get_opc().get_property (OPC_PROP_PINI, tse);
    process_field_numeric (EPICS_DB_PINI, pini);

    // go through properties
    for (property_map::const_iterator f = arg.get_opc().get_properties().begin();
        f != arg.get_opc().get_properties().end(); ++f)
    {
        switch (f->first) {
        case OPC_PROP_UNIT :
            process_field_string (EPICS_DB_EGU, f->second);
            break;
        case OPC_PROP_DESC :
            // processed above
            break;
        case OPC_PROP_HIEU :
            process_field_numeric (EPICS_DB_HOPR, f->second);
            break;
        case OPC_PROP_LOEU :
            process_field_numeric (EPICS_DB_LOPR, f->second);
            break;
        case OPC_PROP_HIRANGE :
            process_field_numeric (EPICS_DB_DRVH, f->second);
            break;
        case OPC_PROP_LORANGE :
            process_field_numeric (EPICS_DB_DRVL, f->second);
            break;
        case OPC_PROP_CLOSE :
            process_field_string (EPICS_DB_ONAM, f->second);
            break;
        case OPC_PROP_OPEN :
            process_field_string (EPICS_DB_ZNAM, f->second);
            break;
        case OPC_PROP_PREC :
            process_field_numeric (EPICS_DB_PREC, f->second);
            break;
        case OPC_PROP_ZRST :
        case OPC_PROP_ZRST + 1 :
        case OPC_PROP_ZRST + 2 :
        case OPC_PROP_ZRST + 3 :
        case OPC_PROP_ZRST + 4 :
        case OPC_PROP_ZRST + 5 :
        case OPC_PROP_ZRST + 6 :
        case OPC_PROP_ZRST + 7 :
        case OPC_PROP_ZRST + 8 :
        case OPC_PROP_ZRST + 9 :
        case OPC_PROP_ZRST + 10 :
        case OPC_PROP_ZRST + 11 :
        case OPC_PROP_ZRST + 12 :
        case OPC_PROP_ZRST + 13 :
        case OPC_PROP_ZRST + 14 :
        case OPC_PROP_FFST :
            process_field_numeric (EPICS_DB_ZRVL[f->first-OPC_PROP_ZRST], f->first-
OPC_PROP_ZRST);
            process_field_string (EPICS_DB_ZRST[f->first-OPC_PROP_ZRST], f->second);
            break;
```

```
        case OPC_PROP_RECTYPE :
        case OPC_PROP_INOUT :
        case OPC_PROP_TSE :
        case OPC_PROP_PINI :
        case OPC_PROP_DTYP :
        case OPC_PROP_SERVER :
        case OPC_PROP_PLCNAME :
            // processed above
            break;
            // alarm
        case OPC_PROP_ALMOSV:
            if ((tname == "bi") || (tname == "bo")) {
                process_field_alarm (EPICS_DB_OSV, f->second);
            }
            break;
        case OPC_PROP_ALMZSV:
            if ((tname == "bi") || (tname == "bo")) {
                process_field_alarm (EPICS_DB_ZSV, f->second);
            }
            break;
        case OPC_PROP_ALMCOSV:
            if ((tname == "bi") || (tname == "bo") || (tname == "mbbi") || (tname == "mbbo")) {
                process_field_alarm (EPICS_DB_COSV, f->second);
            }
            break;
        case OPC_PROP_ALMUNSV:
            if ((tname == "mbbi") || (tname == "mbbo")) {
                process_field_alarm (EPICS_DB_UNSV, f->second);
            }
            break;
        case OPC_PROP_ALMZRSV:
        case OPC_PROP_ALMZRSV + 1:
        case OPC_PROP_ALMZRSV + 2:
        case OPC_PROP_ALMZRSV + 3:
        case OPC_PROP_ALMZRSV + 4:
        case OPC_PROP_ALMZRSV + 5:
        case OPC_PROP_ALMZRSV + 6:
        case OPC_PROP_ALMZRSV + 7:
        case OPC_PROP_ALMZRSV + 8:
        case OPC_PROP_ALMZRSV + 9:
        case OPC_PROP_ALMZRSV + 10:
        case OPC_PROP_ALMZRSV + 11:
        case OPC_PROP_ALMZRSV + 12:
        case OPC_PROP_ALMZRSV + 13:
        case OPC_PROP_ALMZRSV + 14:
        case OPC_PROP_ALMFFSV:
            if ((tname == "mbbi") || (tname == "mbbo")) {
                process_field_alarm (EPICS_DB_UNSV, f->second);
            }
            break;
        case OPC_PROP_ALMHH:
            if ((tname == "ai") || (tname == "ao") || (tname == "longin") || (tname == "longout")) {
                process_field_numeric (EPICS_DB_HIHI, f->second);
                process_field_alarm (EPICS_DB_HHSV, EPICS_DB_MAJOR);
            }
            break;
        case OPC_PROP_ALMH:
            if ((tname == "ai") || (tname == "ao") || (tname == "longin") || (tname == "longout")) {
                process_field_numeric (EPICS_DB_HIGH, f->second);
                process_field_alarm (EPICS_DB_HSV, EPICS_DB_MINOR);
            }
            break;
        case OPC_PROP_ALML:
            if ((tname == "ai") || (tname == "ao") || (tname == "longin") || (tname == "longout")) {
                process_field_numeric (EPICS_DB_LOW, f->second);
                process_field_alarm (EPICS_DB_LSV, EPICS_DB_MINOR);
            }
            break;
        case OPC_PROP_ALMLL:
            if ((tname == "ai") || (tname == "ao") || (tname == "longin") || (tname == "longout")) {
                process_field_numeric (EPICS_DB_LOLO, f->second);
                process_field_alarm (EPICS_DB_LOLO, EPICS_DB_MAJOR);
            }
            break;
        case OPC_PROP_ALMDB:
            if ((tname == "ai") || (tname == "ao") || (tname == "longin") || (tname == "longout")) {
                process_field_numeric (EPICS_DB_HYST, f->second);
            }
            break;
            // unknown
        default:
            if (f->first >= 1000) {
                fprintf (stderr, "Unknown property %i for %s\n", f->first, arg.
    get_name().c_str());
            }
            break;
        }
```

```
    }
    // end with closing bracket
    fprintf(get_file(), "}\n");
    return true;
}
```

Here is the call graph for this function:



8.20.2.2 bool EpicsTpy::epics_db_processing::process_field_alarm (std::stringcase *name*, std::stringcase *severity*) [protected]

Process a record field of type alarm

Parameters

name	Name of field
severity	Severity of alarm

Returns

True if successful

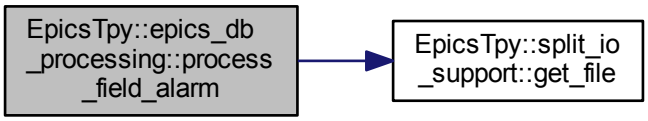
Definition at line 603 of file TpyToEpics.cpp.

References EpicsTpy::split_io_support::get_file().

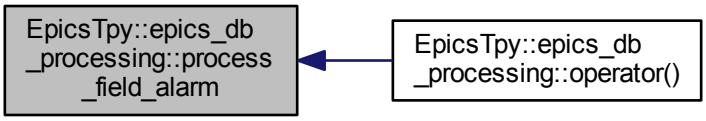
Referenced by operator()().

```
{
    if ((severity == EPICS_DB_NOALARM) || (severity == EPICS_DB_MINOR) ||
        (severity == EPICS_DB_MAJOR)) {
        fprintf (get_file(), "\tfield(%s,\"%s\")\n", name.c_str(), severity.c_str());
        return true;
    }
    else {
        fprintf (stderr, "Unknown alarm severity %s for %s\n", severity.c_str(), name.c_str());
        return false;
    }
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.20.2.3 bool EpicsTpy::epics_db_processing::process_field_numeric (std::stringcase *name*, int *val*) [protected]

Process a record field of numeric type

Parameters

name	Name of field
val	Value of field

Returns

True if successful

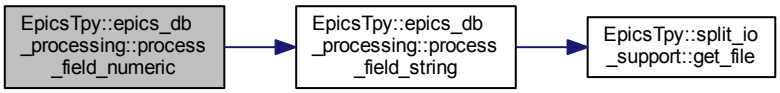
Definition at line 583 of file TpyToEpics.cpp.

References process_field_string().

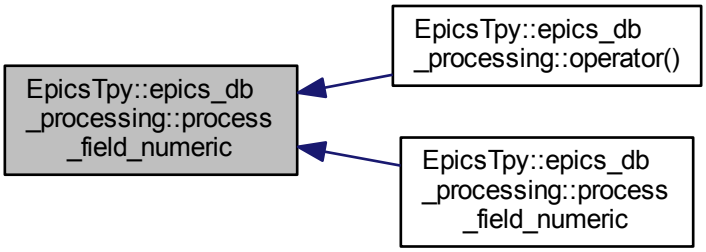
Referenced by operator()(), and process_field_numeric().

```
{
    char buf[40];
    sprintf_s (buf, sizeof(buf), "%i", val);
    return process_field_string (name, buf);
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.20.2.4 bool EpicsTpy::epics_db_processing::process_field_numeric (std::stringcase *name*, std::stringcase *val*) [protected]

Process a record field of type string

Parameters

name	Name of field
val	Value of field

Returns

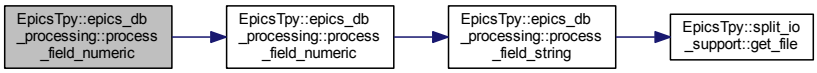
True if successful

Definition at line 593 of file TpyToEpics.cpp.

References process_field_numeric().

```
{
    int v = strtol (val.c_str(), NULL, 10);
    return process_field_numeric (name, v);
}
```

Here is the call graph for this function:



8.20.2.5 bool EpicsTpy::epics_db_processing::process_field_string (std::stringcase *name*, std::stringcase *val*) [protected]

Process a record field of type string

Parameters

name	Name of field
val	Value of field

Returns

True if successful

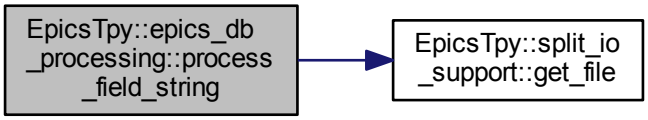
Definition at line 573 of file TpyToEpics.cpp.

References EpicsTpy::split_io_support::get_file().

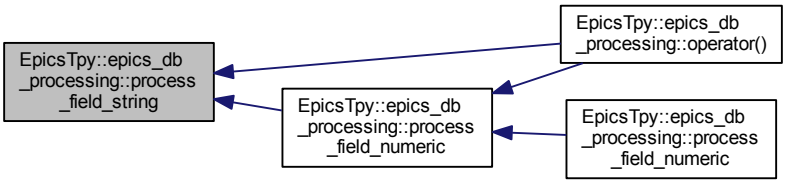
Referenced by operator()(), and process_field_numeric().

```
{
    fprintf (get_file(), "\tfield(%s,\"%s\")\n", name.c_str(), val.c_str());
    return true;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



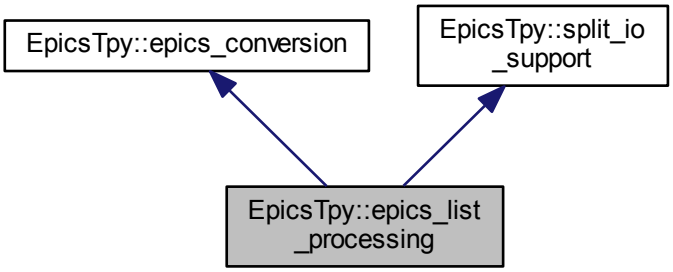
The documentation for this class was generated from the following files:

- TpyToEpics.h
- TpyToEpics.cpp

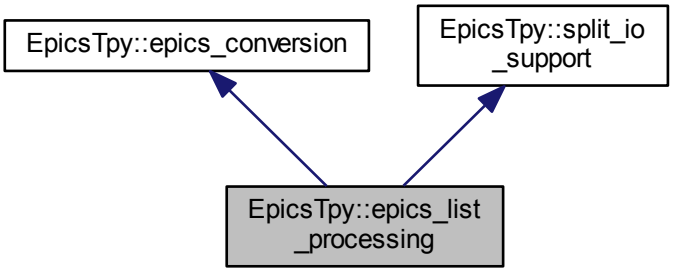
8.21 EpicsTpy::epics_list_processing Class Reference

```
#include <TpyToEpics.h>
```

Inheritance diagram for EpicsTpy::epics_list_processing:



Collaboration diagram for EpicsTpy::epics_list_processing:



Public Member Functions

- epics_list_processing ()

  Default constructor.
- epics_list_processing (listing_type ltype, bool ll=false)
- bool operator() (const ParseTpy::process_arg &arg)
- listing_type get_listing () const

  Get listing type.
- void set_listing (listing_type lt)

  Set listing.
- bool is_verbose () const

  Is long listing?
- void set_verbose (bool vrbs)

  Set long listing.

Protected Attributes

- listing_type listing

  Listing type.
- bool verbose

  long listing

Additional Inherited Members

8.21.1 Detailed Description

Class for generatig a channel list

Definition at line 229 of file TpyToEpics.h.

8.21.2 Constructor & Destructor Documentation

8.21.2.1 EpicsTpy::epics_list_processing::epics_list_processing (listing_type *ltype,* bool *ll* = `false`) [inline], [explicit]

Constructor

Parameters

ltype	Type of listing
ll	long listing

Definition at line 238 of file TpyToEpics.h.

```
240             : listing (listing_standard), verbose (ll) {}
```

8.21.3 Member Function Documentation

8.21.3.1 bool EpicsTpy::epics_list_processing::operator() (const ParseTpy::process_arg & *arg*)

Process a variable

Parameters

arg	Process argument describign the variable and type

Returns

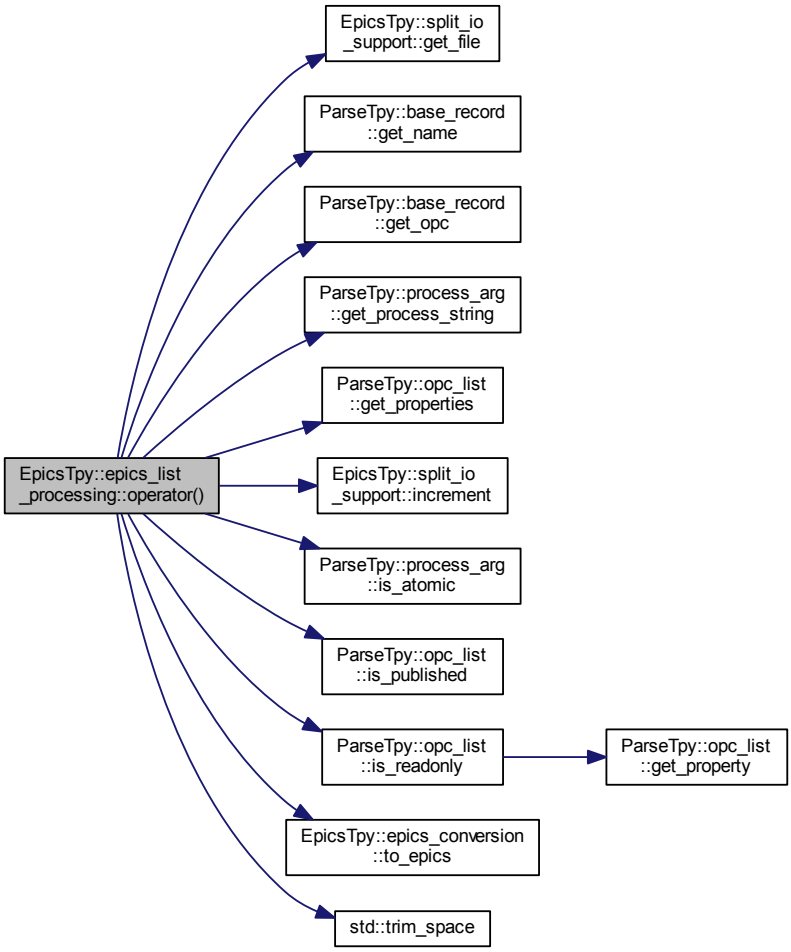
True if successfully processed

Definition at line 288 of file TpyToEpics.cpp.

References EpicsTpy::split_io_support::get_file(), ParseTpy::base_record::get_name(), ParseTpy::base_record::get_opc(), ParseTpy::process_arg::get_process_string(), ParseTpy::opc_list::get_properties(), EpicsTpy::split_io_support::increment(), ParseTpy::process_arg::is_atomic(), ParseTpy::opc_list::is_published(), ParseTpy::opc_list::is_readonly(), listing, EpicsTpy::listing_autoburt, EpicsTpy::listing_epics, EpicsTpy::listing_standard, EpicsTpy::epics_conversion::to_epics(), std::trim_space(), and verbose.

```
289 {
290     // quit if not atomic and not a standard listing
291     if (!arg.is_atomic() && (listing != listing_standard)) {
292         return false;
293     }
294
295     increment (arg.get_opc().is_readonly());
296     // write record information to output file
297     // produce a listing
298     string epicsname = to_epics (arg.get_name());
299     if (listing == listing_epics) {
300         fprintf (get_file(), "%s", epicsname.c_str());
301     }
302     else if (listing == listing_autoburt) {
303         stringcase ro = arg.get_opc().is_readonly() ? "RO " : "";
304         fprintf (get_file(), "%s%s", ro.c_str(), epicsname.c_str());
305     }
306     // standard listing
307     else {
308         fprintf (get_file(), "%s", epicsname.c_str());
309     }
310
311     // long listing?
312     if (verbose && (listing != listing_autoburt)) {
313         fprintf (get_file(), " (%s", arg.get_process_string().c_str());
314         fprintf (get_file(), ", opc %c", arg.get_opc().
      is_published() ? '1' : '0');
315         for (property_map::const_iterator i = arg.get_opc().
      get_properties().begin();
316             i != arg.get_opc().get_properties().end(); ++i) {
317                 stringcase s = i->second;
318                 trim_space (s);
319                 fprintf (get_file(), ", prop[%i]=\"%s\"", i->first, s.c_str());
320         }
321         fprintf (get_file(), ")");
322     }
323     fprintf (get_file(), "\n");
324     return true;
325 }
```

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- TpyToEpics.h
- TpyToEpics.cpp

8.22 DevTc::epics_record_traits< RecType > Struct Template Reference

epics record traits.

```
#include <devTc.h>
```

Classes

- struct traits_type

  Epics record type.

Public Types

- typedef epicsFloat64 value_type

  Value type of (raw) value field.

Static Public Member Functions

- static const char *const name ()

  Name of the record.
- static value_type * val (traits_type *prec)

  Returns the (raw) value of a record.

Static Public Attributes

- static const aitEnum value_ait_type = aitEnumFloat64
- static const aitInt32 value_count = 0
- static const int value_conversion = 0

  return value for read_io functions 0=default, 2=don't convert
- static const bool input_record = true

  Indicates if this is an input record.
- static const bool raw_record = false

  Indicates if this is a raw record.

8.22.1 Detailed Description

template<epics_record_enum RecType>struct DevTc::epics_record_traits< RecType >

epics record traits.

This traits class for Epics records.

Definition at line 167 of file devTc.h.

8.22.2 Member Data Documentation

8.22.2.1 template<epics_record_enum RecType> const aitEnum DevTc::epics_record_traits< RecType >::value_ait_type = aitEnumFloat64 [static]

Data type of records val/rval field aitEnumInvalid type signals an array. Take type/len from record

Definition at line 177 of file devTc.h.

8.22.2.2 template<epics_record_enum RecType> const aitInt32 DevTc::epics_record_traits< RecType >::value_count = 0 [static]

Array length: 1=scalar value, 0=array - see nelm for length, fixed for strings according to the record

Definition at line 180 of file devTc.h.

The documentation for this struct was generated from the following file:

- devTc.h

8.23 DevTc::epics_record_traits< aaival > Struct Template Reference

Epics traits class specialization for aai record.

```
#include <devTcTemplate.h>
```

Public Types

- typedef aaiRecord **traits_type**
- typedef void * **value_type**

Static Public Member Functions

- static const char *const **name** ()
- static value_type * **val** (traits_type *prec)

Static Public Attributes

- static const aitEnum **value_ait_type** = aitEnumFloat64
- static const aitInt32 **value_count** = 0
- static const int **value_conversion** = 0
- static const bool **input_record** = true
- static const bool **raw_record** = false

8.23.1 Detailed Description

template<>struct DevTc::epics_record_traits< aaival >

Epics traits class specialization for aai record.

Definition at line 31 of file devTcTemplate.h.

The documentation for this struct was generated from the following file:

- devTcTemplate.h

8.24 DevTc::epics_record_traits< aaoval > Struct Template Reference

Epics traits class specialization for aao record.

```
#include <devTcTemplate.h>
```

Public Types

- typedef aaoRecord **traits_type**
- typedef void * **value_type**

Static Public Member Functions

- static const char *const **name** ()
- static value_type * **val** (traits_type *prec)

Static Public Attributes

- static const aitEnum **value_ait_type** = aitEnumFloat64
- static const aitInt32 **value_count** = 0
- static const int **value_conversion** = 0
- static const bool **input_record** = false
- static const bool **raw_record** = false

8.24.1 Detailed Description

template<>struct DevTc::epics_record_traits< aaoval >

Epics traits class specialization for aao record.

Definition at line 46 of file devTcTemplate.h.

The documentation for this struct was generated from the following file:

- devTcTemplate.h

8.25 DevTc::epics_record_traits< airval > Struct Template Reference

Epics traits class specialization for ai raw record.

```
#include <devTcTemplate.h>
```

Public Types

- typedef aiRecord **traits_type**
- typedef epicsInt32 **value_type**

Static Public Member Functions

- static const char *const **name** ()
- static value_type * **val** (traits_type *prec)

Static Public Attributes

- static const aitEnum **value_ait_type** = aitEnumInt32
- static const aitInt32 **value_count** = 1
- static const int **value_conversion** = 0
- static const bool **input_record** = true
- static const bool **raw_record** = true

8.25.1 Detailed Description

template<>struct DevTc::epics_record_traits< airval >

Epics traits class specialization for ai raw record.

Definition at line 76 of file devTcTemplate.h.

The documentation for this struct was generated from the following file:

- devTcTemplate.h

8.26 DevTc::epics_record_traits< aival > Struct Template Reference

Epics traits class specialization for ai record.

```
#include <devTcTemplate.h>
```

Public Types

- typedef aiRecord **traits_type**
- typedef epicsFloat64 **value_type**

Static Public Member Functions

- static const char *const **name** ()
- static value_type * **val** (traits_type *prec)

Static Public Attributes

- static const aitEnum **value_ait_type** = aitEnumFloat64
- static const aitInt32 **value_count** = 1
- static const int **value_conversion** = 2
- static const bool **input_record** = true
- static const bool **raw_record** = false

8.26.1 Detailed Description

template<>struct DevTc::epics_record_traits< aival >

Epics traits class specialization for ai record.

Definition at line 61 of file devTcTemplate.h.

The documentation for this struct was generated from the following file:

- devTcTemplate.h

8.27 DevTc::epics_record_traits< aorval > Struct Template Reference

Epics traits class specialization for ao raw record.

```
#include <devTcTemplate.h>
```

Public Types

- typedef aoRecord **traits_type**
- typedef epicsInt32 **value_type**

Static Public Member Functions

- static const char *const **name** ()
- static value_type * **val** (traits_type *prec)

Static Public Attributes

- static const aitEnum **value_ait_type** = aitEnumInt32
- static const aitInt32 **value_count** = 1
- static const int **value_conversion** = 0
- static const bool **input_record** = false
- static const bool **raw_record** = true

8.27.1 Detailed Description

template<>struct DevTc::epics_record_traits< aorval >

Epics traits class specialization for ao raw record.

Definition at line 106 of file devTcTemplate.h.

The documentation for this struct was generated from the following file:

- devTcTemplate.h

8.28 DevTc::epics_record_traits< aoval > Struct Template Reference

Epics traits class specialization for ao record.

```
#include <devTcTemplate.h>
```

Public Types

- typedef aoRecord **traits_type**
- typedef epicsFloat64 **value_type**

Static Public Member Functions

- static const char *const **name** ()
- static value_type * **val** (traits_type *prec)

Static Public Attributes

- static const aitEnum **value_ait_type** = aitEnumFloat64
- static const aitInt32 **value_count** = 1
- static const int **value_conversion** = 0
- static const bool **input_record** = false
- static const bool **raw_record** = false

8.28.1 Detailed Description

template<>struct DevTc::epics_record_traits< aoval >

Epics traits class specialization for ao record.

Definition at line 91 of file devTcTemplate.h.

The documentation for this struct was generated from the following file:

- devTcTemplate.h

8.29 DevTc::epics_record_traits< birval > Struct Template Reference

Epics traits class specialization for bi raw record.

```
#include <devTcTemplate.h>
```

Public Types

- typedef biRecord **traits_type**
- typedef epicsUInt32 **value_type**

Static Public Member Functions

- static const char *const **name** ()
- static value_type * **val** (traits_type *prec)

Static Public Attributes

- static const aitEnum **value_ait_type** = aitEnumUint32
- static const aitInt32 **value_count** = 1
- static const int **value_conversion** = 0
- static const bool **input_record** = true
- static const bool **raw_record** = true

8.29.1 Detailed Description

template<>struct DevTc::epics_record_traits< birval >

Epics traits class specialization for bi raw record.

Definition at line 136 of file devTcTemplate.h.

The documentation for this struct was generated from the following file:

- devTcTemplate.h

8.30 DevTc::epics_record_traits< bival > Struct Template Reference

Epics traits class specialization for bi record.

```
#include <devTcTemplate.h>
```

Public Types

- typedef biRecord **traits_type**
- typedef epicsEnum16 **value_type**

Static Public Member Functions

- static const char *const **name** ()
- static value_type * **val** (traits_type *prec)

Static Public Attributes

- static const aitEnum **value_ait_type** = aitEnumUint16
- static const aitInt32 **value_count** = 1
- static const int **value_conversion** = 2
- static const bool **input_record** = true
- static const bool **raw_record** = false

8.30.1 Detailed Description

template<>struct DevTc::epics_record_traits< bival >

Epics traits class specialization for bi record.

Definition at line 121 of file devTcTemplate.h.

The documentation for this struct was generated from the following file:

- devTcTemplate.h

8.31 DevTc::epics_record_traits< borval > Struct Template Reference

Epics traits class specialization for bo raw record.

```
#include <devTcTemplate.h>
```

Public Types

- typedef boRecord **traits_type**
- typedef epicsUInt32 **value_type**

Static Public Member Functions

- static const char *const **name** ()
- static value_type * **val** (traits_type *prec)

Static Public Attributes

- static const aitEnum **value_ait_type** = aitEnumUint32
- static const aitInt32 **value_count** = 1
- static const int **value_conversion** = 0
- static const bool **input_record** = false
- static const bool **raw_record** = true

8.31.1 Detailed Description

template<>struct DevTc::epics_record_traits< borval >

Epics traits class specialization for bo raw record.

Definition at line 166 of file devTcTemplate.h.

The documentation for this struct was generated from the following file:

- devTcTemplate.h

8.32 DevTc::epics_record_traits< boval > Struct Template Reference

Epics traits class specialization for bo record.

```
#include <devTcTemplate.h>
```

Public Types

- typedef boRecord **traits_type**
- typedef epicsEnum16 **value_type**

Static Public Member Functions

- static const char *const **name** ()
- static value_type * **val** (traits_type *prec)

Static Public Attributes

- static const aitEnum **value_ait_type** = aitEnumUint16
- static const aitInt32 **value_count** = 1
- static const int **value_conversion** = 2
- static const bool **input_record** = false
- static const bool **raw_record** = false

8.32.1 Detailed Description

template<>struct DevTc::epics_record_traits< boval >

Epics traits class specialization for bo record.

Definition at line 151 of file devTcTemplate.h.

The documentation for this struct was generated from the following file:

- devTcTemplate.h

8.33 DevTc::epics_record_traits< eventval > Struct Template Reference

Epics traits class specialization for event record.

```
#include <devTcTemplate.h>
```

Public Types

- typedef eventRecord **traits_type**
- typedef epicsUInt16 **value_type**

Static Public Member Functions

- static const char *const **name** ()
- static value_type * **val** (traits_type *prec)

Static Public Attributes

- static const aitEnum **value_ait_type** = aitEnumUint16
- static const aitInt32 **value_count** = 1
- static const int **value_conversion** = 0
- static const bool **input_record** = true
- static const bool **raw_record** = false

8.33.1 Detailed Description

template<>struct DevTc::epics_record_traits< eventval >

Epics traits class specialization for event record.

Definition at line 376 of file devTcTemplate.h.

The documentation for this struct was generated from the following file:

- devTcTemplate.h

8.34 DevTc::epics_record_traits< histogramval > Struct Template Reference

Epics traits class specialization for histogram record.

```
#include <devTcTemplate.h>
```

Public Types

- typedef histogramRecord **traits_type**
- typedef void * **value_type**

Static Public Member Functions

- static const char *const **name** ()
- static value_type * **val** (traits_type *prec)

Static Public Attributes

- static const aitEnum **value_ait_type** = aitEnumInvalid
- static const aitInt32 **value_count** = 1
- static const int **value_conversion** = 0
- static const bool **input_record** = true
- static const bool **raw_record** = false

8.34.1 Detailed Description

template<>struct DevTc::epics_record_traits< histogramval >

Epics traits class specialization for histogram record.

Definition at line 391 of file devTcTemplate.h.

The documentation for this struct was generated from the following file:

- devTcTemplate.h

8.35 DevTc::epics_record_traits< longinval > Struct Template Reference

Epics traits class specialization for longin record.

```
#include <devTcTemplate.h>
```

Public Types

- typedef longinRecord **traits_type**
- typedef epicsInt32 **value_type**

Static Public Member Functions

- static const char *const **name** ()
- static value_type * **val** (traits_type *prec)

Static Public Attributes

- static const aitEnum **value_ait_type** = aitEnumInt32
- static const aitInt32 **value_count** = 1
- static const int **value_conversion** = 2
- static const bool **input_record** = true
- static const bool **raw_record** = false

8.35.1 Detailed Description

template<>struct DevTc::epics_record_traits< longinval >

Epics traits class specialization for longin record.

Definition at line 181 of file devTcTemplate.h.

The documentation for this struct was generated from the following file:

- devTcTemplate.h

8.36 DevTc::epics_record_traits< longoutval > Struct Template Reference

Epics traits class specialization for longout record.

```
#include <devTcTemplate.h>
```

Public Types

- typedef longoutRecord **traits_type**
- typedef epicsInt32 **value_type**

Static Public Member Functions

- static const char *const **name** ()
- static value_type * **val** (traits_type *prec)

Static Public Attributes

- static const aitEnum **value_ait_type** = aitEnumInt32
- static const aitInt32 **value_count** = 1
- static const int **value_conversion** = 0
- static const bool **input_record** = false
- static const bool **raw_record** = false

8.36.1 Detailed Description

template<>struct DevTc::epics_record_traits< longoutval >

Epics traits class specialization for longout record.

Definition at line 196 of file devTcTemplate.h.

The documentation for this struct was generated from the following file:

- devTcTemplate.h

8.37 DevTc::epics_record_traits< mbbiDirectrval > Struct Template Reference

Epics traits class specialization for mbbiDirect raw record.

```
#include <devTcTemplate.h>
```

Public Types

- typedef mbbiDirectRecord **traits_type**
- typedef epicsUInt32 **value_type**

Static Public Member Functions

- static const char *const **name** ()
- static value_type * **val** (traits_type *prec)

Static Public Attributes

- static const aitEnum **value_ait_type** = aitEnumUint32
- static const aitInt32 **value_count** = 1
- static const int **value_conversion** = 0
- static const bool **input_record** = true
- static const bool **raw_record** = true

8.37.1 Detailed Description

template<>struct DevTc::epics_record_traits< mbbiDirectrval >

Epics traits class specialization for mbbiDirect raw record.

Definition at line 286 of file devTcTemplate.h.

The documentation for this struct was generated from the following file:

- devTcTemplate.h

8.38 DevTc::epics_record_traits< mbbiDirectval > Struct Template Reference

Epics traits class specialization for mbbiDirect record.

```
#include <devTcTemplate.h>
```

Public Types

- typedef mbbiDirectRecord **traits_type**
- typedef epicsEnum16 **value_type**

Static Public Member Functions

- static const char *const **name** ()
- static value_type * **val** (traits_type *prec)

Static Public Attributes

- static const aitEnum **value_ait_type** = aitEnumUint16
- static const aitInt32 **value_count** = 1
- static const int **value_conversion** = 2
- static const bool **input_record** = true
- static const bool **raw_record** = false

8.38.1 Detailed Description

template<>struct DevTc::epics_record_traits< mbbiDirectval >

Epics traits class specialization for mbbiDirect record.

Definition at line 271 of file devTcTemplate.h.

The documentation for this struct was generated from the following file:

- devTcTemplate.h

8.39 DevTc::epics_record_traits< mbbirval > Struct Template Reference

Epics traits class specialization for mbbi raw record.

```
#include <devTcTemplate.h>
```

Public Types

- typedef mbbiRecord **traits_type**
- typedef epicsUInt32 **value_type**

Static Public Member Functions

- static const char *const **name** ()
- static value_type * **val** (traits_type *prec)

Static Public Attributes

- static const aitEnum **value_ait_type** = aitEnumUint32
- static const aitInt32 **value_count** = 1
- static const int **value_conversion** = 0
- static const bool **input_record** = true
- static const bool **raw_record** = true

8.39.1 Detailed Description

template<>struct DevTc::epics_record_traits< mbbirval >

Epics traits class specialization for mbbi raw record.

Definition at line 226 of file devTcTemplate.h.

The documentation for this struct was generated from the following file:

- devTcTemplate.h

8.40 DevTc::epics_record_traits< mbbival > Struct Template Reference

Epics traits class specialization for mbbi record.

```
#include <devTcTemplate.h>
```

Public Types

- typedef mbbiRecord **traits_type**
- typedef epicsEnum16 **value_type**

Static Public Member Functions

- static const char *const **name** ()
- static value_type * **val** (traits_type *prec)

Static Public Attributes

- static const aitEnum **value_ait_type** = aitEnumUint16
- static const aitInt32 **value_count** = 1
- static const int **value_conversion** = 2
- static const bool **input_record** = true
- static const bool **raw_record** = false

8.40.1 Detailed Description

template<>struct DevTc::epics_record_traits< mbbival >

Epics traits class specialization for mbbi record.

Definition at line 211 of file devTcTemplate.h.

The documentation for this struct was generated from the following file:

- devTcTemplate.h

8.41 DevTc::epics_record_traits< mbboDirectrval > Struct Template Reference

Epics traits class specialization for mbboDirect raw record.

```
#include <devTcTemplate.h>
```

Public Types

- typedef mbboDirectRecord **traits_type**
- typedef epicsUInt32 **value_type**

Static Public Member Functions

- static const char *const **name** ()
- static value_type * **val** (traits_type *prec)

Static Public Attributes

- static const aitEnum **value_ait_type** = aitEnumUint32
- static const aitInt32 **value_count** = 1
- static const int **value_conversion** = 0
- static const bool **input_record** = false
- static const bool **raw_record** = true

8.41.1 Detailed Description

template<>struct DevTc::epics_record_traits< mbboDirectrval >

Epics traits class specialization for mbboDirect raw record.

Definition at line 316 of file devTcTemplate.h.

The documentation for this struct was generated from the following file:

- devTcTemplate.h

8.42 DevTc::epics_record_traits< mbboDirectval > Struct Template Reference

Epics traits class specialization for mbboDirect record.

```
#include <devTcTemplate.h>
```

Public Types

- typedef mbboDirectRecord **traits_type**
- typedef epicsEnum16 **value_type**

Static Public Member Functions

- static const char *const **name** ()
- static value_type * **val** (traits_type *prec)

Static Public Attributes

- static const aitEnum **value_ait_type** = aitEnumUint16
- static const aitInt32 **value_count** = 1
- static const int **value_conversion** = 0
- static const bool **input_record** = false
- static const bool **raw_record** = false

8.42.1 Detailed Description

template<>struct DevTc::epics_record_traits< mbboDirectval >

Epics traits class specialization for mbboDirect record.

Definition at line 301 of file devTcTemplate.h.

The documentation for this struct was generated from the following file:

- devTcTemplate.h

8.43 DevTc::epics_record_traits< mbborval > Struct Template Reference

Epics traits class specialization for mbbo raw record.

```
#include <devTcTemplate.h>
```

Public Types

- typedef mbboRecord **traits_type**
- typedef epicsUInt32 **value_type**

Static Public Member Functions

- static const char *const **name** ()
- static value_type * **val** (traits_type *prec)

Static Public Attributes

- static const aitEnum **value_ait_type** = aitEnumUint32
- static const aitInt32 **value_count** = 1
- static const int **value_conversion** = 0
- static const bool **input_record** = false
- static const bool **raw_record** = true

8.43.1 Detailed Description

template<>struct DevTc::epics_record_traits< mbborval >

Epics traits class specialization for mbbo raw record.

Definition at line 256 of file devTcTemplate.h.

The documentation for this struct was generated from the following file:

- devTcTemplate.h

8.44 DevTc::epics_record_traits< mbboval > Struct Template Reference

Epics traits class specialization for mbbo record.

```
#include <devTcTemplate.h>
```

Public Types

- typedef mbboRecord **traits_type**
- typedef epicsEnum16 **value_type**

Static Public Member Functions

- static const char *const **name** ()
- static value_type * **val** (traits_type *prec)

Static Public Attributes

- static const aitEnum **value_ait_type** = aitEnumUint16
- static const aitInt32 **value_count** = 1
- static const int **value_conversion** = 0
- static const bool **input_record** = false
- static const bool **raw_record** = false

8.44.1 Detailed Description

template<>struct DevTc::epics_record_traits< mbboval >

Epics traits class specialization for mbbo record.

Definition at line 241 of file devTcTemplate.h.

The documentation for this struct was generated from the following file:

- devTcTemplate.h

8.45 DevTc::epics_record_traits< stringinval > Struct Template Reference

Epics traits class specialization for stringin record.

```
#include <devTcTemplate.h>
```

Public Types

- typedef stringinRecord **traits_type**
- typedef char **value_type** [40]

Static Public Member Functions

- static const char *const **name** ()
- static value_type * **val** (traits_type *prec)

Static Public Attributes

- static const aitEnum **value_ait_type** = aitEnumString
- static const aitInt32 **value_count** = 40
- static const int **value_conversion** = 0
- static const bool **input_record** = true
- static const bool **raw_record** = false

8.45.1 Detailed Description

template<>struct DevTc::epics_record_traits< stringinval >

Epics traits class specialization for stringin record.

Definition at line 331 of file devTcTemplate.h.

The documentation for this struct was generated from the following file:

- devTcTemplate.h

8.46 DevTc::epics_record_traits< stringoutval > Struct Template Reference

Epics traits class specialization for stringout record.

```
#include <devTcTemplate.h>
```

Public Types

- typedef stringoutRecord **traits_type**
- typedef char **value_type** [40]

Static Public Member Functions

- static const char *const **name** ()
- static value_type * **val** (traits_type *prec)

Static Public Attributes

- static const aitEnum **value_ait_type** = aitEnumString
- static const aitInt32 **value_count** = 40
- static const int **value_conversion** = 0
- static const bool **input_record** = false
- static const bool **raw_record** = false

8.46.1 Detailed Description

template<>struct DevTc::epics_record_traits< stringoutval >

Epics traits class specialization for stringout record.

Definition at line 346 of file devTcTemplate.h.

The documentation for this struct was generated from the following file:

- devTcTemplate.h

8.47 DevTc::epics_record_traits< waveformval > Struct Template Reference

Epics traits class specialization for waveform record.

```
#include <devTcTemplate.h>
```

Public Types

- typedef waveformRecord **traits_type**
- typedef void * **value_type**

Static Public Member Functions

- static const char *const **name** ()
- static value_type * **val** (traits_type *prec)

Static Public Attributes

- static const aitEnum **value_ait_type** = aitEnumInvalid
- static const aitInt32 **value_count** = 0
- static const int **value_conversion** = 0
- static const bool **input_record** = true
- static const bool **raw_record** = false

8.47.1 Detailed Description

template<>struct DevTc::epics_record_traits< waveformval >

Epics traits class specialization for waveform record.

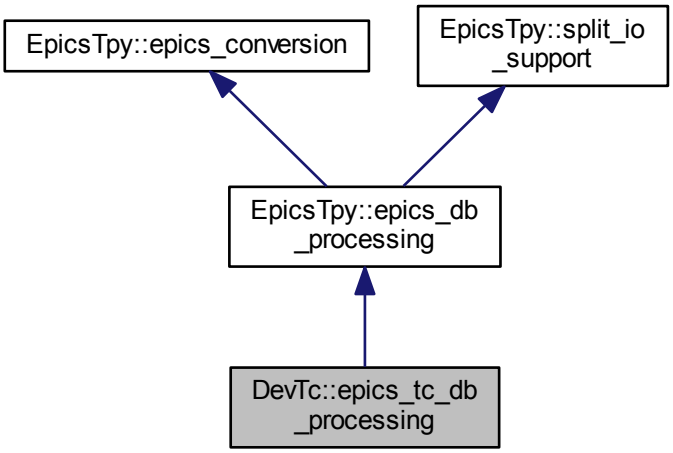
Definition at line 361 of file devTcTemplate.h.

The documentation for this struct was generated from the following file:

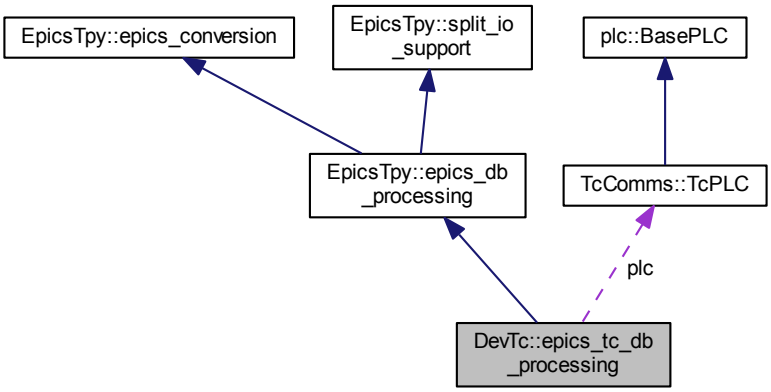
- devTcTemplate.h

8.48 DevTc::epics_tc_db_processing Class Reference

Inheritance diagram for DevTc::epics_tc_db_processing:



Collaboration diagram for DevTc::epics_tc_db_processing:



Public Member Functions

- epics_tc_db_processing (TcComms::TcPLC &p)

  Default constructor.
- bool operator() (const ParseTpy::process_arg &arg)
- int get_invalid_records () const

  Get number of EPICS records without tc records.

Protected Attributes

- TcComms::TcPLC * plc

  Pointer to PLC class.
- int invnum

  Number of EPICS records without tc records.

Additional Inherited Members

8.48.1 Detailed Description

Definition at line 56 of file drvTc.cpp.

8.48.2 Member Function Documentation

8.48.2.1 bool DevTc::epics_tc_db_processing::operator() (const ParseTpy::process_arg & *arg*)

Process a variable

Parameters

arg	Process argument describign the variable and type

Returns

True if successful

Determine data type of this record

Make new record object

Make TCat interface

Tell TCat interface about its parent PLC

Tell record about TCat interface

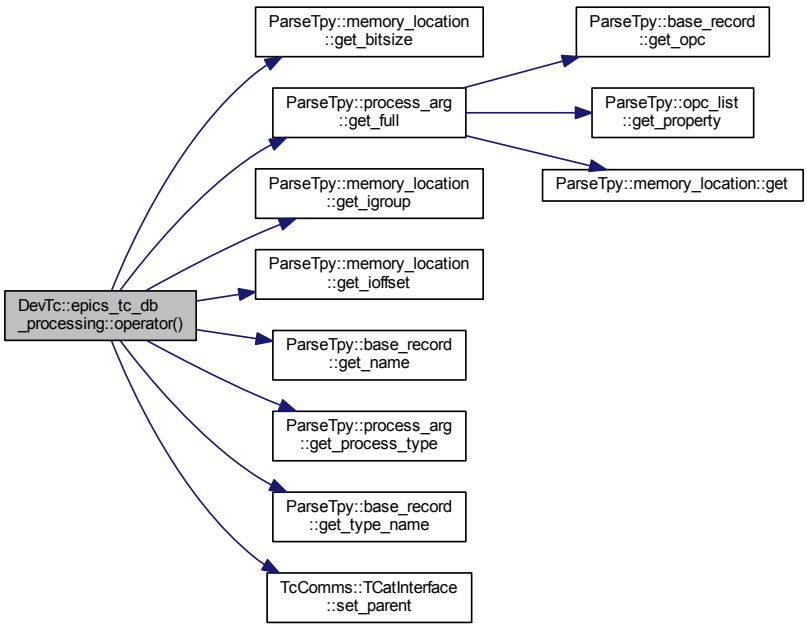
Definition at line 81 of file drvTc.cpp.

References plc::dtBool, plc::dtDouble, plc::dtFloat, plc::dtInt16, plc::dtInt32, plc::dtInt8, plc::dtInvalid, plc::dtString, plc::dtUInt16, plc::dtUInt32, plc::dtUInt8, ParseTpy::memory_location::get_bitsize(), ParseTpy::process_arg::get_full(), ParseTpy::memory_location::get_igroup(), ParseTpy::memory_location::get_ioffset(), ParseTpy::base_record::get_name(), ParseTpy::process_arg::get_process_type(), ParseTpy::base_record::get_type_name(), ParseTpy::pt_binary, ParseTpy::pt_enum, and TcComms::TCatInterface::set_parent().

```
{
    // Generate EPICS database record
    if (!EpicsTpy::epics_db_processing::operator()(arg)) {
        return false;
    }
    // add tc record
    if (arg.get_bitsize() % 8 != 0) {
        ++invnum;
        return false;
    }

    plc::data_type_enum rt = plc::data_type_enum::dtInvalid;

    if (arg.get_type_name() == "BOOL") rt = plc::data_type_enum::dtBool;
    else if (arg.get_type_name() == "SINT") rt = plc::data_type_enum::dtInt8;
    else if (arg.get_type_name() == "USINT" || arg.get_type_name() == "BYTE") rt = plc::data_type_enum::dtUInt8;
    else if (arg.get_type_name() == "INT"|| arg.get_process_type() == ParseTpy::pt_enum) rt = plc::data_type_enum::dtInt16;
```

```
    else if (arg.get_type_name() == "UINT" || arg.get_type_name() =="WORD") rt = plc::data_type_enum::dtUInt16;
    else if (arg.get_type_name() == "DINT") rt = plc::data_type_enum::dtInt32;
    else if (arg.get_type_name() == "UDINT" || arg.get_type_name() == "DWORD") rt = plc::data_type_enum::dtUInt32;
    else if (arg.get_type_name() == "REAL") rt = plc::data_type_enum::dtFloat;
    else if (arg.get_type_name() == "LREAL") rt = plc::data_type_enum::dtDouble;
    else if (arg.get_type_name().substr(0,6) == "STRING") rt = plc::data_type_enum::dtString;

    plc::BaseRecordPtr pRecord = plc::BaseRecordPtr(new plc::BaseRecord(arg.get_full(),rt));
    TcComms::TCatInterface* tcat = new TcComms::TCatInterface (*pRecord,
                                arg.get_name(),
                                arg.get_igroup(),
                                arg.get_ioffset(),
                                arg.get_bitsize() / 8,
                                arg.get_type_name(),
                                arg.get_process_type() == ParseTpy::pt_binary,
                                arg.get_process_type() == ParseTpy::pt_enum);
    //tcat->set_pRecord(pRecord.get());
    tcat->set_parent(plc);
    pRecord->set_plcInterface(tcat);

    if (!plc->add(pRecord)) {
        // this means the EPICS record has nothing to connect to!
        ++invnum;
        return false;
    }

    return true;
}
```

Here is the call graph for this function:



The documentation for this class was generated from the following file:

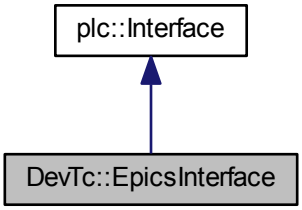
- drvTc.cpp

8.49 DevTc::EpicsInterface Class Reference

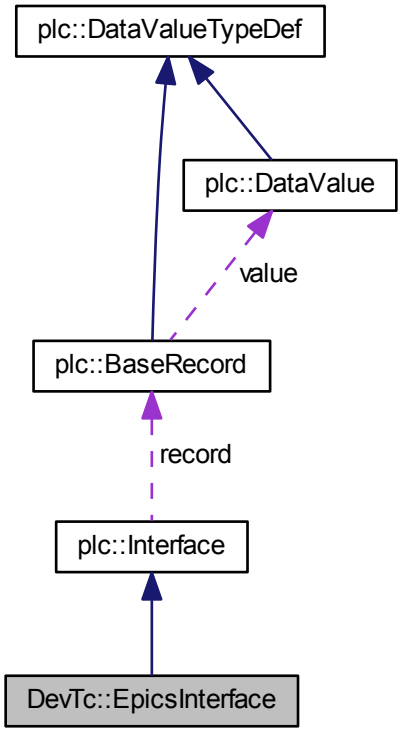
epics interface class.

```
#include <devTc.h>
```

Inheritance diagram for DevTc::EpicsInterface:



Collaboration diagram for DevTc::EpicsInterface:



Public Member Functions

- EpicsInterface (plc::BaseRecord &dval)

  Constructor.
- ~EpicsInterface ()

  Deconstructor.
- void set_ioscanpvt (IOSCANPVT ioscan)

  Set ioscanpvt.
- void set_isPassive (bool passive)

  Set isPassive.
- bool get_isCallback ()

  Get isCallback.
- void set_isCallback (bool isCb)

  Set isCallback.
- void set_pEpicsRecord (dbCommon *pEpRecord)

  Set pEpicsRecord.
- void * get_pEpicsVal ()

  Get pEpicsVal.
- void set_pEpicsVal (void *pVal)

  Set pEpicsVal.
- const unsigned long get_size ()

  Get size.
- void set_size (unsigned long nBytes)

  Set size.
- bool get_callbackRequestPending ()

  Get callbackRequestPending.
- void set_callbackRequestPending (bool pending)

  Set callbackRequestPending.
- virtual bool push () override

  Makes a call to the EPICS dbProcess function.
- virtual bool pull () override

  Does nothing.

Public Attributes

- IOSCANPVT **ioscanpvt**
- CALLBACK **callback**

Protected Attributes

- bool isPassive
- bool isCallback
- dbCommon * pEpicsRecord

  Pointer to the EPICS record.
- void * pEpicsVal

  Pointer to the R field of the EPICS record.
- unsigned long size

  Size (bytes) of the data.
- bool callbackRequestPending

8.49.1 Detailed Description

epics interface class.

device support for TwinCAT/ADSThis is a class for an EPICS Interface

Definition at line 31 of file devTc.h.

8.49.2 Member Function Documentation

8.49.2.1 bool DevTc::EpicsInterface::pull ( ) [override], [virtual]

Does nothing.

EpicsInterface::pull

Implements plc::Interface.

Definition at line 115 of file devTc.cpp.

```
116 {
117     return true;
118 }
```

8.49.2.2 bool DevTc::EpicsInterface::push ( ) [override], [virtual]

Makes a call to the EPICS dbProcess function.

EpicsInterface::push

Implements plc::Interface.

Definition at line 97 of file devTc.cpp.

```
98 {
99     if (isCallback) {
100
101         // Generate IO intr request
102         if (isPassive) {
103             callbackRequestPending = true;
104             callbackRequest(&callback);
105         }
106         else {
107             scanIoRequest( ioscanpvt );
108         }
109     }
110     return true;
111 }
```

8.49.3 Member Data Documentation

8.49.3.1 bool DevTc::EpicsInterface::callbackRequestPending [protected]

Bool indicating that a read callback is pending Set to true for in/out records when callback request is made. true : if this is an in/out record, then dbProcess will do an EPICS read instead of write, then reset this value to false

Definition at line 98 of file devTc.h.

Referenced by get_callbackRequestPending(), and set_callbackRequestPending().

8.49.3.2 bool DevTc::EpicsInterface::isCallback [protected]

Bool indicating whether callback is needed to call dbProcess true : SCAN = I/O Intr or the record is an out record

Definition at line 87 of file devTc.h.

Referenced by get_isCallback(), and set_isCallback().

8.49.3.3 bool DevTc::EpicsInterface::isPassive [protected]

Bool indicating passive scan true : EPICS record SCAN field is set to PASSIVE

Definition at line 84 of file devTc.h.

Referenced by set_isPassive().

The documentation for this class was generated from the following files:

- devTc.h
- devTc.cpp

8.50 std::std::hash< stringcase > Struct Template Reference

hash for case insensitive string.

`#include <stringcase_hash.h>`

Public Member Functions

- std::size_t operator() (const stringcase &str) const

8.50.1 Detailed Description

template<>struct std::std::hash< stringcase >

hash for case insensitive string.

This is a function specialization for case sensitive strings. Perform a 32/64 bit Fowler/Noll/Vo hash on a string.

Definition at line 11 of file stringcase_hash.h.

8.50.2 Member Function Documentation

8.50.2.1 std::size_t std::std::hash< stringcase >::operator() (const stringcase & *str*) const

Perform a 32 or 64 bit Fowler/Noll/Vo hash on a string Adapted from fnv_32_str and fnv_64_str http://www.-isthe.com/chongo/tech/comp/fnv/index.html#FNV-source

Parameters

std	string to hash

Returns

32/64 bit hash

The documentation for this struct was generated from the following file:

- stringcase_hash.h

8.51 std::std::hash< wstringcase > Struct Template Reference

hash for case insensitive unicode string.

`#include <stringcase_hash.h>`

Public Member Functions

- std::size_t operator() (const wstringcase &str) const

8.51.1 Detailed Description

template<>struct std::std::hash< wstringcase >

hash for case insensitive unicode string.

This is a function specialization for case sensitive unicode strings. Perform a 32/64 bit Fowler/Noll/Vo hash on a unicode string.

Definition at line 27 of file stringcase_hash.h.

8.51.2 Member Function Documentation

8.51.2.1 std::size_t std::std::hash< wstringcase >::operator() (const wstringcase & *str*) const

Perform a 32 or 64 bit Fowler/Noll/Vo hash on a string Adapted from fnv_32_str http://www.isthe.-com/chongo/tech/comp/fnv/index.html#FNV-source

Parameters

std	string to hash

Returns

32/64 bit hash

The documentation for this struct was generated from the following file:

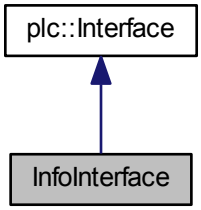
- stringcase_hash.h

8.52 InfoInterface Class Reference

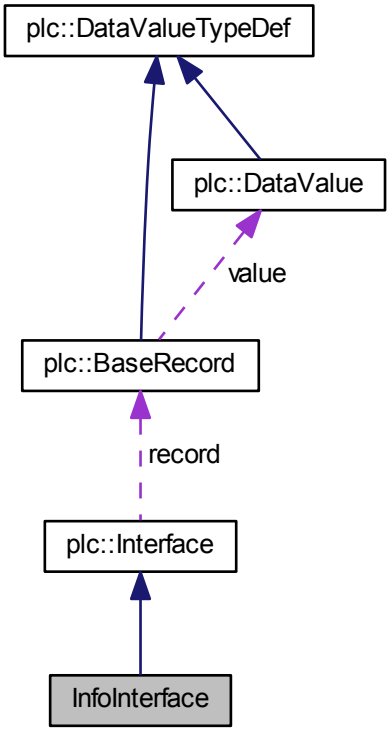
Info interface.

`#include <infoPlc.h>`

Inheritance diagram for InfoInterface:



Collaboration diagram for InfoInterface:



Public Member Functions

- **InfoInterface** (plc::BaseRecord &dval)
- virtual bool push () override

  Pure virtual method indicating that the value needs to be pushed.
- virtual bool pull () override

  Pure virtual method indicating that the value needs to be pulled.

Additional Inherited Members

8.52.1 Detailed Description

Info interface.

This is a class for a Info interface

Definition at line 12 of file infoPlc.h.

The documentation for this class was generated from the following file:

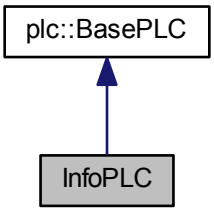
- infoPlc.h

8.53 InfoPLC Class Reference

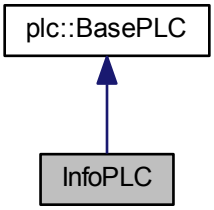
Info plc.

`#include <infoPlc.h>`

Inheritance diagram for InfoPLC:



Collaboration diagram for InfoPLC:



Public Member Functions

- InfoPLC ()
- ~InfoPLC ()

Additional Inherited Members

8.53.1 Detailed Description

Info plc.

This is a class for a dummy PLC that tracks connection status, errors, etc.

Definition at line 27 of file infoPlc.h.

8.53.2 Constructor & Destructor Documentation

8.53.2.1 InfoPLC::InfoPLC ( )

constructor

8.53.2.2 InfoPLC::~InfoPLC ( )

destructor

The documentation for this class was generated from the following file:

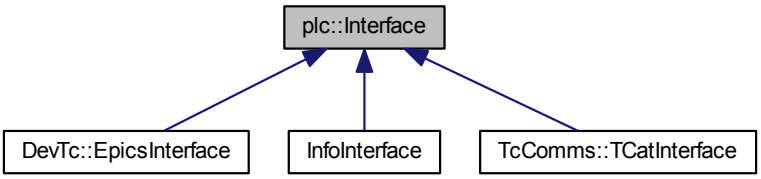
- infoPlc.h

8.54 plc::Interface Class Reference

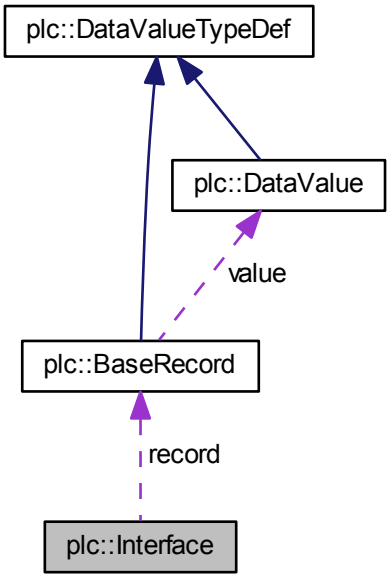
abstract interface

```
#include <plcBase.h>
```

Inheritance diagram for plc::Interface:



Collaboration diagram for plc::Interface:



Public Member Functions

- Interface (BaseRecord &dval)
- virtual ~Interface ()

  Desctructor.
- BaseRecord * get_record ()

  Return a pointer to the tag/channel record.
- void set_record (BaseRecord &rec)

  Set the tag/channel record reference.
- virtual bool push ()=0

  Pure virtual method indicating that the value needs to be pushed.
- virtual bool pull ()=0

  Pure virtual method indicating that the value needs to be pulled.

Protected Attributes

- BaseRecord * record

  Pointer to tag/channel record associated with this interface.

8.54.1 Detailed Description

abstract interface

This is a base class for an abstract interface to access the PLC (slave) or the user side (master).

Definition at line 25 of file plcBase.h.

8.54.2 Constructor & Destructor Documentation

8.54.2.1 plc::Interface::Interface (BaseRecord & *dval*) [inline],[explicit]

Constructor

Parameters

dval	Reference to a tag/channel record

Definition at line 30 of file plcBase.h.

```
30 : record (&dval) {}
```

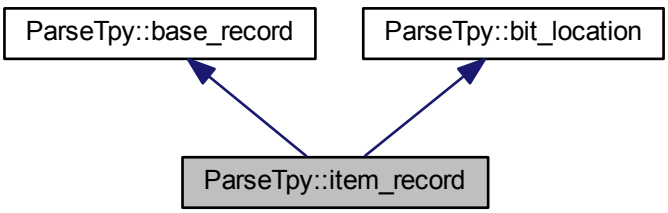
The documentation for this class was generated from the following file:

- plcBase.h

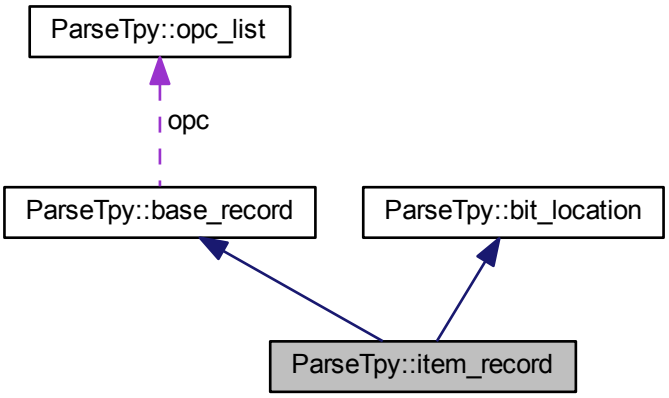
8.55 ParseTpy::item_record Class Reference

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::item_record:



Collaboration diagram for ParseTpy::item_record:



Public Member Functions

- item_record ()

  Default constructor.

Additional Inherited Members

8.55.1 Detailed Description

This class stores typed items.

Definition at line 268 of file ParseTpy.h.

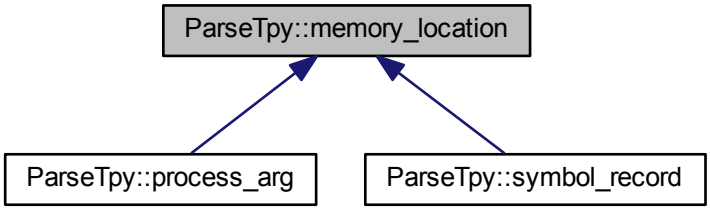
The documentation for this class was generated from the following file:

- ParseTpy.h

8.56 ParseTpy::memory_location Class Reference

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::memory_location:



Public Member Functions

- memory_location ()

  Default constructor.
- memory_location (int ig, int io, int bs)

  Constructor.
- memory_location (const std::stringcase &s)

  Constructor.
- bool isValid () const

  Validity.
- int get_igroup () const

  Get IGroup.
- void set_igroup (int ig)

  Set IGroup.
- int get_ioffset () const

  Get IOffset.
- void set_ioffset (int io)

  Set IOffset.
- int get_bitsize () const

  Get BitSize.
- void set_bitsize (int bs)

  Set BitSize.
- bool set_section (const bit_location &loc)
- std::stringcase get () const
- bool set (const std::stringcase &s)

Protected Attributes

- int igroup

  Memory group.
- int ioffset

  Memory offset.
- int bitsize

  Memory size in bits.

8.56.1 Detailed Description

This structure holds a memory location

Definition at line 359 of file ParseTpy.h.

8.56.2 Member Function Documentation

8.56.2.1 std::stringcase ParseTpy::memory_location::get ( ) const

Gets a string representation of a memory location

Returns

string with format "igroup/ioffset:size", empty on error

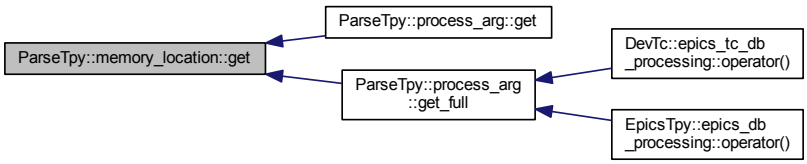
Definition at line 376 of file ParseTpy.cpp.

References bitsize, igroup, and ioffset.

Referenced by ParseTpy::process_arg::get(), and ParseTpy::process_arg::get_full().

```
377 {
378     if ((igroup == -1) || (ioffset == -1) ||
379         (bitsize <= 0) || (bitsize % 8 != 0)) {
380         return "";
381     }
382     char buf[256];
383     sprintf_s (buf, sizeof (buf), "%i/%i:%i", igroup, ioffset,
      bitsize / 8);
384     buf[255] = 0;
385     return buf;
386 }
```

Here is the caller graph for this function:



8.56.2.2 bool ParseTpy::memory_location::set (const std::stringcase & *s*)

Set the memory location using a string of the form: "igroup/ioffset:size" where the size is in bytes

Parameters

s	String describing memory location

Returns

True if successful

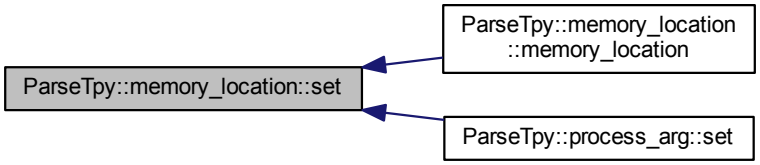
Definition at line 390 of file ParseTpy.cpp.

References bitsize, igroup, and ioffset.

Referenced by memory_location(), and ParseTpy::process_arg::set().

```
{
    int ig (0), io (0), sz (0), n (0);
    int num = sscanf_s (s.c_str(), "%i/%i:%i%n", ig, io, sz, &n);
    if ((num != 3) || (n != s.length())) {
        igroup = -1;
        ioffset = -1;
        bitsize = -1;
        return false;
    }
    igroup = ig;
    ioffset = io;
    bitsize = 8 * sz;
    return true;
}
```

Here is the caller graph for this function:



8.56.2.3 bool ParseTpy::memory_location::set_section (const bit_location & *loc*)

Set a sub section

Parameters

loc	Bit location witin memory region

Returns

True if section within bounds, false otherwise

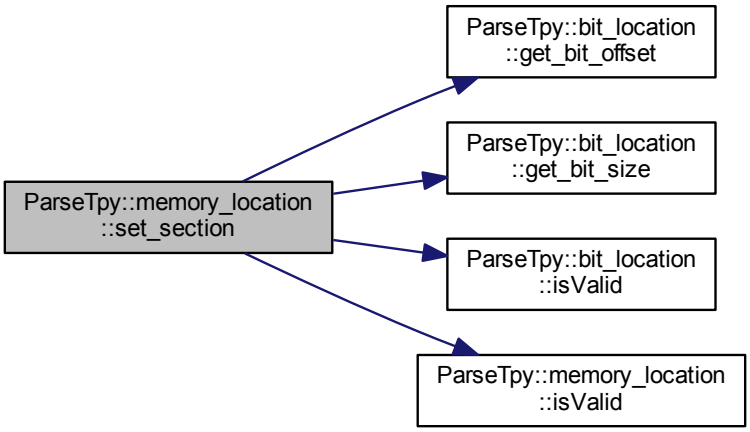
Definition at line 361 of file ParseTpy.cpp.

References bitsize, ParseTpy::bit_location::get_bit_offset(), ParseTpy::bit_location::get_bit_size(), ioffset, ParseTpy::bit_location::isValid(), and isValid().

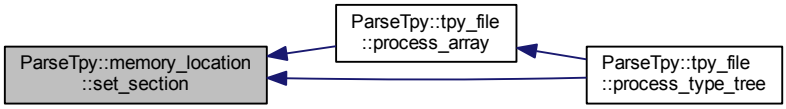
Referenced by ParseTpy::tpy_file::process_array(), and ParseTpy::tpy_file::process_type_tree().

```
{
    if (!loc.isValid() || !isValid() ||
        (loc.get_bit_offset() % 8 != 0) ||
        (loc.get_bit_size() % 8 != 0) ||
        (loc.get_bit_offset() + loc.get_bit_size() > bitsize)) {
            return false;
    }
    ioffset += loc.get_bit_offset() / 8;
    bitsize = loc.get_bit_size();
    return true;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- ParseTpy.h
- ParseTpy.cpp

8.57 ParseTpy::opc_list Class Reference

```
#include <ParseTpy.h>
```

Public Member Functions

- opc_list ()

  Default constructor.
- opc_enum get_opc_state () const

  Get opc state.
- void set_opc_state (opc_enum state)

  Set opc state.
- const property_map & get_properties () const

  Get opc property list.

- property_map & get_properties ()

  Get opc property list.
- const property_map::const_iterator get_property (int key) const

  Get the specified property.
- void add (const property_el &el)

  Add an OPC property.
- void add (const opc_list &o)

  Add an OPC list.
- bool is_published () const

  Is this item published?
- bool is_readonly () const

  Is readonly?
- bool get_property (int prop, std::stringcase &val) const

  Get string property.
- bool get_property (int prop, int &val) const

  Get integer property.
- bool get_property (int prop, double &val) const

  Get real property.

Protected Attributes

- opc_enum opc

  OPC state.
- property_map opc_prop

  List of OPC properties.

8.57.1 Detailed Description

This class stores OPC properties.

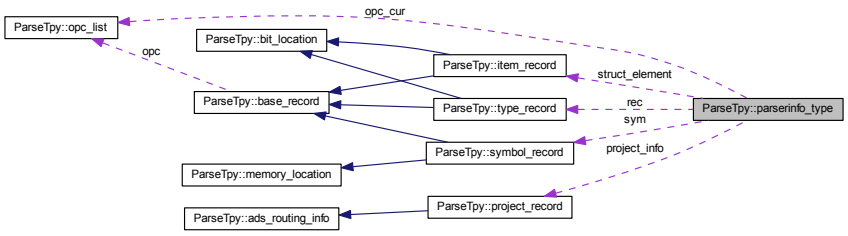
Definition at line 40 of file ParseTpy.h.

The documentation for this class was generated from the following files:

- ParseTpy.h
- ParseTpy.cpp

8.58 ParseTpy::parserinfo_type Class Reference

Collaboration diagram for ParseTpy::parserinfo_type:



Public Member Functions

- **parserinfo_type** (project_record &p, symbol_list &s, type_map &t)
- symbol_list & **get_symbols** ()
- type_map & **get_types** ()
- project_record & **get_projectinfo** ()
- void **init** ()
- type_enum **get_type_description** () const
- bool **verytop** ()
- bool **top** ()

Public Attributes

- int **ignore**
- bool **projects**
- int **routing**
- int **types**
- int **symbols**
- symbol_record **sym**
- type_record **rec**
- int name_parse
- int type_parse

  level indicator for type parsing
- opc_list * opc_cur

  Pointer to current opc list (types only)
- property_el opc_prop

  temporary opc element
- int opc_parse

  level indicator for opc element
- std::stringcase opc_data

  temporary data string for parsed opc data
- int opc_cdata

  temporary cdata indicator
- int igroup_parse

  level indicator for IGroup
- int ioffset_parse

  level indicator for IOffset
- int bitsize_parse

  level indicator for BitSize
- int bitoffs_parse

  level indicator for BitOffs
- std::stringcase data

  temporary data string for parsed igroup/ioffset/bitsize/bitoffs
- int array_parse

  level indicator for array info parsing
- std::stringcase array_data

  temporary data string for parsed array info
- dimension array_bounds

  temporary array dimension element
- int enum_parse

  level indicator for enum parsing
- std::stringcase enum_data

temporary data string for parsed enum data

- enum_pair enum_element

  temporary enum element

- int struct_parse

  level indicator for struct parsing

- item_record struct_element

  temporary structure element

- int fb_parse

  level indicator for function block parsing

Protected Attributes

- symbol_list * **sym_list**
- type_map * **type_list**
- project_record * **project_info**

8.58.1 Detailed Description

Definition at line 38 of file ParseTpy.cpp.

8.58.2 Member Data Documentation

8.58.2.1 int ParseTpy::parserinfo_type::name_parse

level indicator for type name parsing level indicators in general: 0 - not encountered, 1 - parsed, 2 - currently processing, 3 - further sub tag parsing

Definition at line 92 of file ParseTpy.cpp.

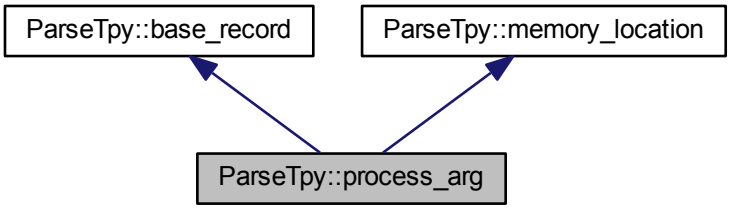
The documentation for this class was generated from the following file:

- ParseTpy.cpp

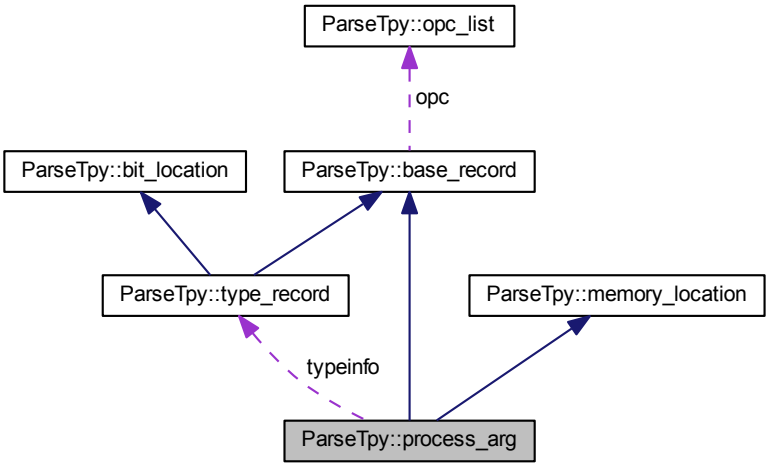
8.59 ParseTpy::process_arg Class Reference

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::process_arg:



Collaboration diagram for ParseTpy::process_arg:



Public Member Functions

- process_arg ()

  Default constructor.
- process_arg (const std::stringcase &name, process_type_enum pt, const opc_list &o, const std::stringcase &tname, int tdecor=0, bool at=true, const type_record *ti=0)
- process_arg (memory_location &loc, const std::stringcase &name, process_type_enum pt, const opc_list &o, const std::stringcase &tname, int tdecor=0, bool at=true, const type_record *ti=0)
- process_type_enum get_process_type () const

  Get process type.
- std::stringcase get_process_string () const

  Get process type string.
- void set_process_type (process_type_enum pt)

  Set process type.
- bool is_atomic () const

  Is atomic (or structured) type.
- void set_atomic (bool at)

  Set atomic type.
- const type_record * get_type_info () const

  Get process type (for structured types, otherwise 0)
- void set_type_info (const type_record *ti)

  Set process type.
- std::stringcase get () const
- bool set (const std::stringcase &s)
- std::stringcase get_full () const

Protected Attributes

- process_type_enum ptype

  Process Type.

- bool atomic

  Atomic element.

- const type_record ∗ typeinfo

  Pointer to type when structured (not owned by this class)

8.59.1 Detailed Description

Argument which is passed to the name/tag processing function.

Definition at line 456 of file ParseTpy.h.

8.59.2 Constructor & Destructor Documentation

8.59.2.1 ParseTpy::process_arg::process_arg (const std::stringcase & *name*, process_type_enum *pt*, const opc_list & *o*, const std::stringcase & *tname*, int *tdecor* = 0, bool *at* = true, const type_record ∗ *ti* = 0) [inline]

Constructor

Parameters

name	Variable name
pt	Process type
o	OPC list
tname	Type name
tdecor	Type decoraton
at	Atomic type
ti	Pointer to type record

Definition at line 470 of file ParseTpy.h.

```
473         : base_record (name, o, tname, tdecor),
474         ptype (pt), atomic (at), typeinfo (ti) {}
```

8.59.2.2 ParseTpy::process_arg::process_arg (memory_location & *loc*, const std::stringcase & *name*, process_type_enum *pt*, const opc_list & *o*, const std::stringcase & *tname*, int *tdecor* = 0, bool *at* = true, const type_record ∗ *ti* = 0) [inline]

Constructor

Parameters

loc	Memory location
name	Variable name
pt	Process type
o	OPC list
tname	Type name
tdecor	Type decoraton
at	Atomic type
ti	Pointer to type record

Definition at line 484 of file ParseTpy.h.

```
488         : base_record (name, o, tname, tdecor), memory_location (loc),
489         ptype (pt), atomic (at), typeinfo (ti) {}
```

8.59.3 Member Function Documentation

8.59.3.1 std::stringcase ParseTpy::process_arg::get ( ) const [inline]

Gets a string representation of a memory location

Returns

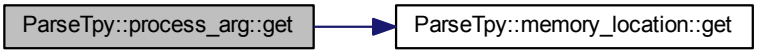
string with format "igroup/ioffset:size", empty on error

Definition at line 508 of file ParseTpy.h.

References ParseTpy::memory_location::get().

```
508                                 {
509         return memory_location::get(); }
```

Here is the call graph for this function:



8.59.3.2 std::stringcase ParseTpy::process_arg::get_full ( ) const

Gets a string representation of a PLC & memory location

Returns

string with format "prefixigroup/ioffset:size", empty on error

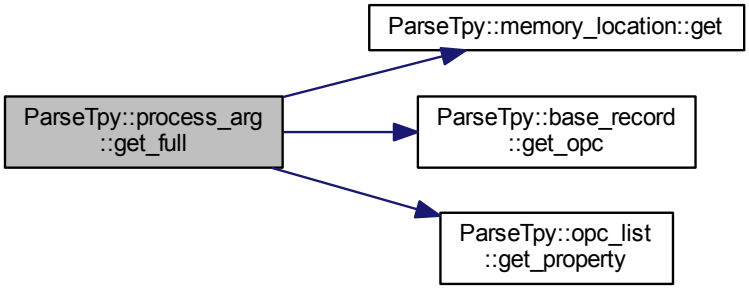
Definition at line 432 of file ParseTpy.cpp.

References ParseTpy::memory_location::get(), ParseTpy::base_record::get_opc(), ParseTpy::opc_list::get_-property(), and ParseTpy::base_record::name.

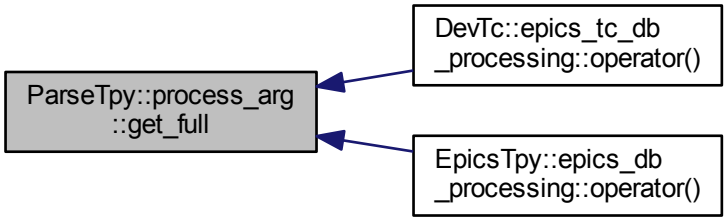
Referenced by DevTc::epics_tc_db_processing::operator()(), and EpicsTpy::epics_db_processing::operator()().

```
433 {
434     std::stringcase servername = "tc://0.0.0.0.0.0:801/";
435     get_opc().get_property (OPC_PROP_PLCNAME, servername);
436     std::stringcase name = memory_location::get();
437     if (!name.empty()) {
438         return servername + name;
439     }
440     else {
441         return "";
442     }
443 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.59.3.3 bool ParseTpy::process_arg::set (const std::stringcase & *s*) [inline]

Set the memory location using a string of the form: "igroup/ioffset:size" where the size is in bytes

Parameters

s	String describing memory location

Returns

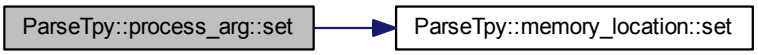
True if successful

Definition at line 514 of file ParseTpy.h.

References ParseTpy::memory_location::set().

```
514                                     {
515         return memory_location::set (s); }
```

Here is the call graph for this function:



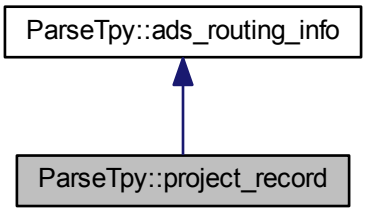
The documentation for this class was generated from the following files:

- ParseTpy.h
- ParseTpy.cpp

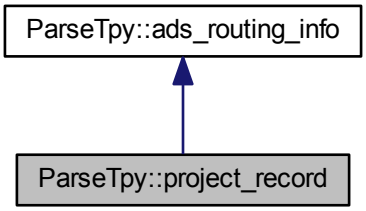
8.60 ParseTpy::project_record Class Reference

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::project_record:



Collaboration diagram for ParseTpy::project_record:



Public Member Functions

- project_record ()

  Default constructor.

Additional Inherited Members

8.60.1 Detailed Description

This is a base class for storing name, type, type id and opc list

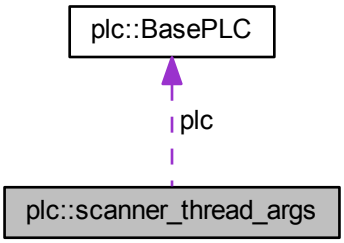
Definition at line 141 of file ParseTpy.h.

The documentation for this class was generated from the following file:

- ParseTpy.h

8.61 plc::scanner_thread_args Struct Reference

Collaboration diagram for plc::scanner_thread_args:



Public Attributes

- plc::BasePLC * **plc**
- long **scanperiod**
- plc::BasePLC::scanner_func **scanner**

8.61.1 Detailed Description

Definition at line 577 of file plcBase.cpp.

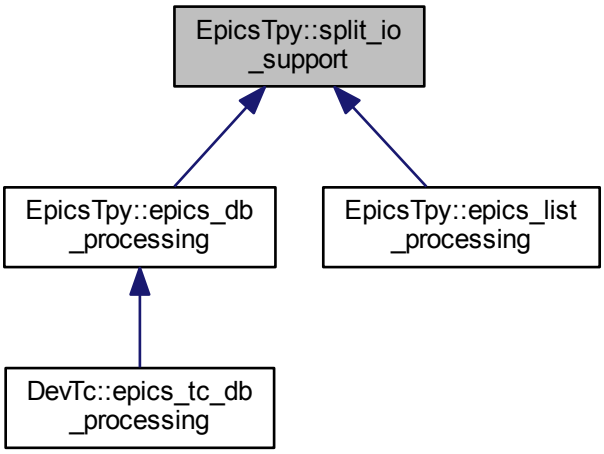
The documentation for this struct was generated from the following file:

- plcBase.cpp

8.62 EpicsTpy::split_io_support Class Reference

```
#include <TpyToEpics.h>
```

Inheritance diagram for EpicsTpy::split_io_support:



Public Member Functions

- split_io_support ()

  Default constructor.
- split_io_support (const std::stringcase &fname, bool split=false, int max=0)

  Default constructor.
- ~split_io_support ()

  Destructor.
- split_io_support (const split_io_support &)
- split_io_support & operator= (const split_io_support &)
- bool operator! () const

  Return error.
- bool increment (bool readonly)
- void flush ()

  Flush contents of output files.
- FILE * get_file () const

  Get output file.
- const std::stringcase & get_filename () const

  Get output filename.
- bool is_split () const

  Is output split?
- int get_max () const

  Maximum of channels per file.
- int get_processed_total () const

  Get number of processed channels.
- int get_processed_readonly () const

Get number of processed readonly channels.

- int get_processed_io () const

  Get number of processed input/ouput channels.

Protected Member Functions

- void set_filename (const std::stringcase &fname)

  Set output filename.

- void close ()

  Close files.

Protected Attributes

- bool error

  Error.

- std::stringcase outfilename

  Output filename.

- bool split_io

  Split output into read only channels and input/output channels.

- int split_n

  Maximum number of channels per file; 0 indicates no limit.

- FILE * outf

  Output file.

- FILE * outf_in

  Output file for read only channels.

- FILE * outf_io

  Output file for input/output channels.

- int rec_num

  Current number of processed channels (records)

- int rec_num_in

  Current number of processed read only channels (records)

- int rec_num_io

  Current number of processed input/output channels (records)

- int file_num_in

  Current file number of processed read only channels (records)

- int file_num_io

  Current file number of processed input/output channels (records)

- std::stringcase file_num_in_s

  Contains the readonly file number in string format.

- std::stringcase file_num_io_s

  Contains the input/output file number in string format.

- std::stringcase file_in_s

  Contains the file extenstion for readonly files.

- std::stringcase file_io_s

  Contains the file extenstion for input/output files.

8.62.1 Detailed Description

Split file IO support Output can be split in multiple files if the number of channels exceeds the maximum specified for a file

Definition at line 113 of file TpyToEpics.h.

8.62.2 Constructor & Destructor Documentation

8.62.2.1 EpicsTpy::split_io_support::split_io_support (const split_io_support & *iosup*)

Copy constructor File pointers will be moved over and the original ones will become invalid.

Definition at line 92 of file TpyToEpics.cpp.

```
93     : outf(0), outf_io (0), outf_in (0)
94 {
95     *this = iosup;
96 }
```

8.62.3 Member Function Documentation

8.62.3.1 bool EpicsTpy::split_io_support::increment (bool *readonly*)

Increase the channel number

Parameters

readonly	Inidcates if channel is readonly

Returns

True if no error

Definition at line 214 of file TpyToEpics.cpp.

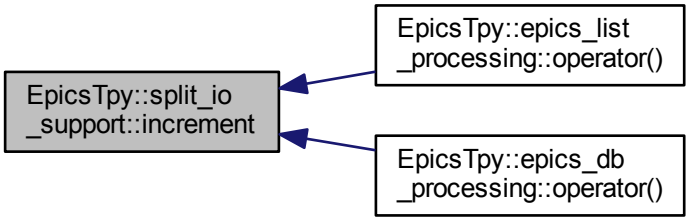
References error, file_in_s, file_io_s, file_num_in, file_num_in_s, file_num_io, file_num_io_s, outf, outf_in, outf_io, outfilename, rec_num, rec_num_in, rec_num_io, split_io, and split_n.

Referenced by EpicsTpy::epics_list_processing::operator()(), and EpicsTpy::epics_db_processing::operator()().

```
215 {
216     // check if we need to open a new split file
217     if (!error && (split_n > 0)) {
218         if (split_io) {
219             if (readonly) {
220                 if ((rec_num_in > 0) && (rec_num_in %
    split_n == 0) && outf_in) {
221                     ++file_num_in;
222                     char buf[20];
223                     sprintf_s (buf, sizeof(buf), ".%03i", file_num_in);
224                     file_num_in_s = buf;
225                     fclose (outf_in);
226                     stringcase fname = outfilename +
    file_in_s + file_num_in_s + ".db";
227                     outf_in = fopen (fname.c_str(), "w");
228                     if (!outf_in) {
229                         fprintf (stderr, "Failed to open output %s.\n", fname.c_str());
230                         error = true;
231                     }
232                 }
233             }
234             else {
235                 if ((rec_num_io > 0) && (rec_num_io %
    split_n == 0) && outf_io) {
236                     ++file_num_io;
237                     char buf[20];
238                     sprintf_s (buf, sizeof(buf), ".%03i", file_num_io);
239                     file_num_io_s = buf;
240                     fclose (outf_io);
241                     stringcase fname (outfilename +
    file_io_s + file_num_io_s + ".db");
242                     outf_io = fopen (fname.c_str(), "w");
243                     if (!outf_io) {
244                         fprintf (stderr, "Failed to open output %s.\n", fname.c_str());
245                         error = true;
246                     }
247                 }
248             }
249         }
250         // no splitting
```

```
        else {
            if ((rec_num > 0) && (rec_num % split_n == 0) && outf_io) {
                ++file_num_io;
                char buf[20];
                sprintf (buf, ".%03i", file_num_io);
                file_num_io_s = buf;
                fclose (outf_io);
                stringcase fname (outfilename + file_io_s + file_num_io_s + ".db");
                outf_io = fopen (fname.c_str(), "w");
                if (!outf_io) {
                    fprintf (stderr, "Failed to open output %s.\n", fname.c_str());
                    error = true;
                }
            }
        }
    }
    // set up output file
    if (split_io && readonly) outf = outf_in;
    else if (!outfilename.empty()) outf = outf_io;
    if (!outf) {
        outf = stdout;
        error = true;
    }
    // increase record number
    if (!error) {
        if (readonly) ++rec_num_in;
        else ++rec_num_io;
        ++rec_num;
    }

    return error;
}
```

Here is the caller graph for this function:



8.62.3.2 split_io_support & EpicsTpy::split_io_support::operator= (const split_io_support & *iosup*)

Assignment File pointers will be moved over andt the original ones will become invalid.

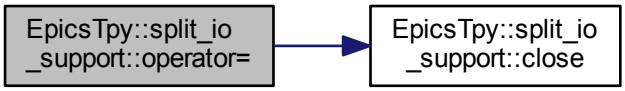
Definition at line 101 of file TpyToEpics.cpp.

References close(), error, file_in_s, file_io_s, file_num_in, file_num_in_s, file_num_io, file_num_io_s, outf, outf_in, outf_io, outfilename, rec_num, rec_num_in, rec_num_io, split_io, and split_n.

```
{
    close();
    error = iosup.error;
    outfilename = iosup.outfilename;
    split_io = iosup.split_io;
    split_n = iosup.split_n;
    rec_num = iosup.rec_num;
    rec_num_in = iosup.rec_num_in;
    rec_num_io = iosup.rec_num_io;
    file_num_in = iosup.file_num_in;
    file_num_io = iosup.file_num_io;
```

```
    file_num_in_s = iosup.file_num_in_s;
    file_num_io_s = iosup.file_num_io_s;
    file_in_s = iosup.file_in_s;
    file_io_s = iosup.file_io_s;
    outf = iosup.outf;
    outf_in = iosup.outf_in;
    outf_io = iosup.outf_io;
    iosup.outf = nullptr;
    iosup.outf_in = nullptr;
    iosup.outf_io = nullptr;
    return *this;
}
```

Here is the call graph for this function:



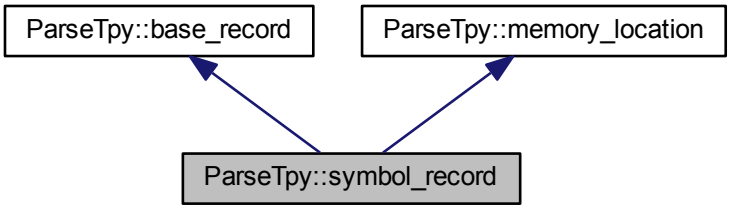
The documentation for this class was generated from the following files:

- TpyToEpics.h
- TpyToEpics.cpp

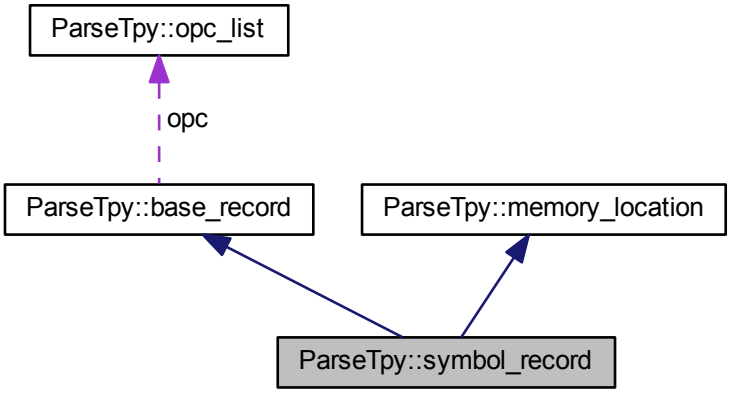
8.63 ParseTpy::symbol_record Class Reference

`#include <ParseTpy.h>`

Inheritance diagram for ParseTpy::symbol_record:



Collaboration diagram for ParseTpy::symbol_record:



Public Member Functions

- symbol_record ()

  Default constructor.
- const memory_location & get_location () const

  get memory location
- memory_location & get_location ()

  get memory location

Additional Inherited Members

8.63.1 Detailed Description

This structure holds a symbol record

Definition at line 412 of file ParseTpy.h.

The documentation for this class was generated from the following file:

- ParseTpy.h

8.64 syminfo_processing Class Reference

Public Member Functions

- **syminfo_processing** (FILE ∗outfile=0, bool atomic=true)
- bool **operator()** (const process_arg &arg)

Protected Attributes

- FILE ∗ **outf**
- bool **firstline**

8.64.1 Detailed Description

Symbol processing

Definition at line 15 of file ParseTpyInfo.cpp.

The documentation for this class was generated from the following file:

- ParseTpyInfo.cpp

8.65 plc::System Class Reference

System.

```
#include <plcBase.h>
```

Public Types

- typedef std::recursive_mutex mutex_type

  Defines the mutex type.
- typedef std::lock_guard < mutex_type > guard

  Defined the mutex guard type.

Public Member Functions

- bool add (BasePLC *plc)
- bool add (BasePLCPtr plc)
- BasePLCPtr find (std::stringcase id)

  Finds a PLC by its name.
- void printVals ()

  Print all PLC record values to the console.

Static Public Member Functions

- static System & get ()

  Return a reference to the gloabl System variable.

Protected Attributes

- mutex_type mux

  Mutex to synchronize access to this class.
- BasePLCList PLCs

  Master list of all PLCs.

8.65.1 Detailed Description

System.

This is a class for managing multiple PLCs.

Definition at line 762 of file plcBase.h.

8.65.2 Member Function Documentation

8.65.2.1 bool plc::System::add (BasePLC * *plc*) [inline]

Add a new PLC, PLC will be adopted

Parameters

Pointer	to plc

Returns

true if successful

Definition at line 775 of file plcBase.h.

```
775                                 {
776         return plc ? add (BasePLCPtr (plc)) : false; }
```

8.65.2.2 bool plc::System::add (BasePLCPtr *plc*)

Add a new PLC

Parameters

Smart	pointer to plc

Returns

true if successful

Definition at line 688 of file plcBase.cpp.

References mux, and PLCs.

```
689 {
690     if (!newPLC.get()) {
691         return false;
692     }
693     // TODO: Error: check if this PLC already exists
694     guard lock (mux);
695     auto ret = PLCs.insert (
696         BasePLCList::value_type (newPLC->get_name(), newPLC));
697     return ret.second;
698 }
```

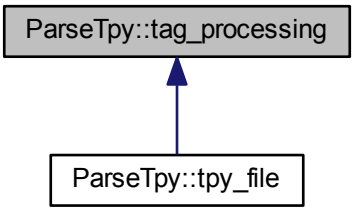
The documentation for this class was generated from the following files:

- plcBase.h
- plcBase.cpp

8.66 ParseTpy::tag_processing Class Reference

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::tag_processing:



Public Member Functions

- tag_processing ()

  Default constructor.
- tag_processing (bool all, process_tag_enum proctags, bool nostring=false)
- tag_processing (int argc, const char ∗const argv[])
- int getopt (int argc, const char ∗const argv[], bool argp[]=0)
- bool get_export_all () const

  Get the export rule.
- void set_export_all (bool all)

  Set the export rule.
- process_tag_enum get_process_tags () const

  Get the process rule.
- void set_process_tags (process_tag_enum proctags)

  Set the process rule.
- bool get_no_strings () const

  Get the string rule.
- void set_no_strings (bool nostring)

  Set the string rule.

Protected Attributes

- bool export_all

  Process all symbols regarless of opc publish setting.
- process_tag_enum process_tags

  Process only atomic types.
- bool no_string_tags

  Don't process strings.

8.66.1 Detailed Description

Class to specify which symbols and tags/names to process

Definition at line 543 of file ParseTpy.h.

8.66.2 Constructor & Destructor Documentation

8.66.2.1 ParseTpy::tag_processing::tag_processing (bool *all*, process_tag_enum *proctags*, bool *nostring* = `false`) `[inline]`

Constructor

Parameters

all	Process all tags
proctags	Process atomic and/or structured tags
nostring	Don't process string tags

Definition at line 553 of file ParseTpy.h.

```
554         : export_all (all), process_tags (proctags),
555         no_string_tags (nostring) {}
```

8.66.2.2 ParseTpy::tag_processing::tag_processing (int *argc*, const char *const *argv[]*) [inline]

Constructor Commaline arguments will override default parameters when specified The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored

Parameters

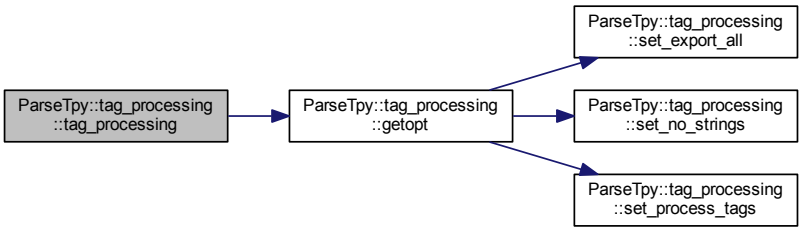
argc	Number of command line arguments
argv	List of command line arguments, same format as in main()

Definition at line 563 of file ParseTpy.h.

References getopt().

```
564         : export_all (false), process_tags (process_all),
565         no_string_tags (false) { getopt (argc, argv); }
```

Here is the call graph for this function:



8.66.3 Member Function Documentation

8.66.3.1 int ParseTpy::tag_processing::getopt (int *argc*, const char *const *argv[]*, bool *argp[]* = 0)

Parse a command line The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored The argp boolean array can be used to pass in a list of already processed (and to be ignored) command line arguments. This list, if supplied, must be at least argc long. Upon return, newly processed arguments are also marked as processed in this list. The arguments are: /ea: Export all variables regardless of OPC setting (default) /eo: Only export variables which are marked as OPC export /ns: No string variables are processed /ys: String variables are processes (default) /pa: Call process for all types (default) /ps: Call process for simple (atomic) types only /pc: Call process for complex (structure and array) types only Command line arguments can use '-' instead of a '/'. Capitalization does not matter. getopt will only override arguments that are specifically specified. It relies on the contructors to provide the defaults.

Parameters

argc	Number of command line arguments
argv	List of command line arguments, same format as in main()
argp	Excluded/processed arguments (in/out), array length must be argc

Returns

Number of arguments processed

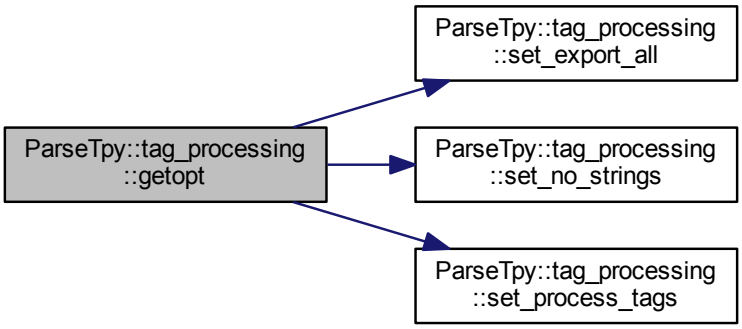
Definition at line 448 of file ParseTpy.cpp.

References ParseTpy::process_all, ParseTpy::process_atomic, ParseTpy::process_structured, set_export_all(), set_no_strings(), and set_process_tags().

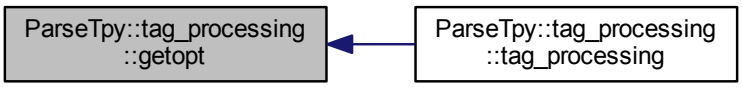
Referenced by tag_processing().

```
{
    int num = 0;
    for (int i = 1; i < argc; ++i) {
        if (!argp || argp[i]) continue;
        if (!argv[i]) {
            argp[i] = true;
            continue;
        }
        std::stringcase arg (argv[i]);
        int oldnum = num;
        // export only opc variables (default)
        if (arg == "-eo" || arg == "/eo" ) {
            set_export_all (false);
            ++num;
        }
        // export all
        else if (arg == "-ea" || arg == "/ea" ) {
            set_export_all (true);
            ++num;
        }
        // Exclude string types (default)
        else if (arg == "-ns" || arg == "/ns") {
            set_no_strings (true);
            ++num;
        }
        // Include string types
        else if (arg == "-ys" || arg == "/ys") {
            set_no_strings (false);
            ++num;
        }
        // Call process for all types (default)
        else if (arg == "-pa" || arg == "/pa") {
            set_process_tags (process_all);
            ++num;
        }
        // Call process for simple types only
        else if (arg == "-ps" || arg == "/ps") {
            set_process_tags (process_atomic);
            ++num;
        }
        // Call process for complex types only
        else if (arg == "-pc" || arg == "/pc") {
            set_process_tags (process_structured);
            ++num;
        }
        // no set flag to indicated a processed option
        if (argp && (num > oldnum)) {
            argp[i] = true;
        }
    }
    return num;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following files:

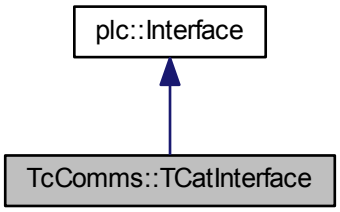
- ParseTpy.h
- ParseTpy.cpp

8.67 TcComms::TCatInterface Class Reference

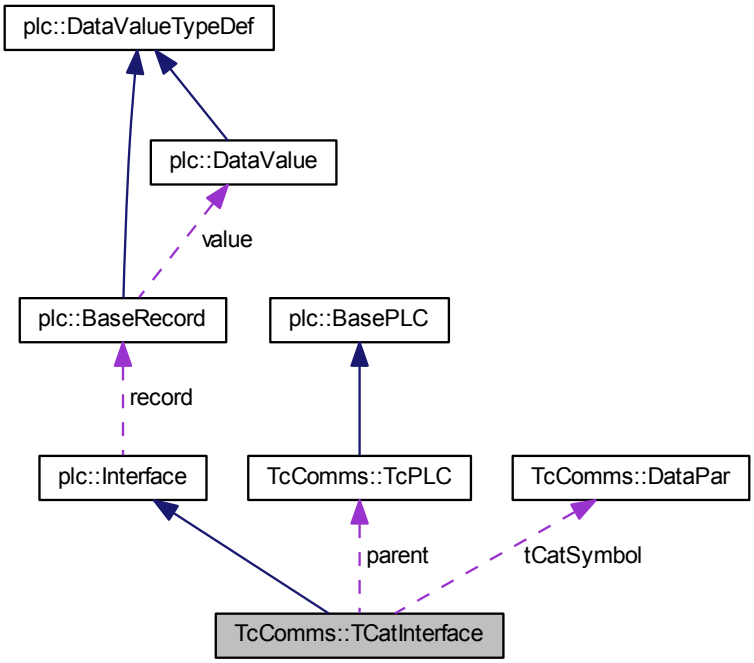
TCat interface class.

```
#include <tcComms.h>
```

Inheritance diagram for TcComms::TCatInterface:



Collaboration diagram for TcComms::TCatInterface:



Public Member Functions

- TCatInterface (plc::BaseRecord &dval)

  Constructor.
- TCatInterface (plc::BaseRecord &dval, const std::stringcase &name, unsigned long group, unsigned long offset, unsigned long length, const std::stringcase &type, bool isStruct, bool isEnum)
- ~TCatInterface ()

  Deconstructor.
- const std::stringcase get_tCatName ()

Get name of TCat symbol.

- void set_tCatName (std::stringcase name)

  Set name of TCat symbol.
- const std::stringcase get_tCatType ()

  Get TCat data type.
- void set_tCatType (std::stringcase type)

  Set TCat data type.
- DataPar get_tCatSymbol ()

  Get structure containing index group, index offset, size.
- unsigned long get_indexGroup () const

  Get index group.
- void set_indexGroup (unsigned long group)

  Set index group.
- unsigned long get_indexOffset () const

  Get index offset.
- void set_indexOffset (unsigned long offset)

  Set index offset.
- unsigned long get_size () const

  Get size in bytes of symbol.
- void set_size (unsigned long nBytes)

  Set size in bytes of symbol.
- size_t get_requestOffs () const

  Get offset into response buffer.
- void set_requestOffs (size_t pVal)

  Set offset into response buffer.
- TcPLC * get_parent ()

  Get parent PLC that owns this record.
- const TcPLC * get_parent () const

  Get parent PLC that owns this record.
- void set_parent (TcPLC *pParent)

  Set pointer to parent PLC that owns this record.
- int get_requestNum ()

  Get the request group number this record is in.
- void set_requestNum (int rNum)

  Set the request group number this record is in.
- void printTCatVal (FILE *fp)
- virtual bool push () override

  Does nothing.
- virtual bool pull () override

  Does nothing.

Protected Attributes

- std::stringcase tCatName

  Name of TCat symbol.
- std::stringcase tCatType

  Data type in TCat.
- DataPar tCatSymbol

  Struct storing index group, index offset, and length of TC symbol.
- TcPLC * parent

Pointer to PLC that owns this.

- int requestNum

  Which request group in the PLC.

- size_t requestOffs

  Offset into response buffer.

8.67.1 Detailed Description

TCat interface class.

This is a class for a TCat interface

Definition at line 74 of file tcComms.h.

8.67.2 Constructor & Destructor Documentation

8.67.2.1 TcComms::TCatInterface::TCatInterface (plc::BaseRecord & *dval,* const std::stringcase & *name,* unsigned long *group,* unsigned long *offset,* unsigned long *length,* const std::stringcase & *type,* bool *isStruct,* bool *isEnum*)

Constructor

Parameters

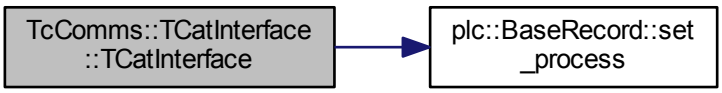
dval	BaseRecord that this interface is part of
name	Name of TCat symbol
group	Index group of TCat symbol
offset	Index offset of TCat symbol
length	Size in bytes of TCat symbol
type	Name of TCat data type
isStruct	True = this symbol is a structure in TCat
isEnum	True = this symbol is an enum in TCat

Definition at line 381 of file tcComms.cpp.

References TcComms::DataPar::indexGroup, TcComms::DataPar::indexOffset, TcComms::DataPar::length, plc::-Interface::record, plc::BaseRecord::set_process(), tCatSymbol, and tCatType.

```
    : Interface (dval), tCatName(name), tCatType(type)
{
    tCatSymbol.indexGroup = group;
    tCatSymbol.indexOffset = offset;
    tCatSymbol.length = nBytes;
    if (isEnum) tCatType = "ENUM";
    if (isStruct) record->set_process(false);
}
```

Here is the call graph for this function:



8.67.3 Member Function Documentation

8.67.3.1 void TcComms::TCatInterface::printTCatVal (FILE * *fp*)

Prints TCat symbol value and information

Parameters

fp	File to print symbol to

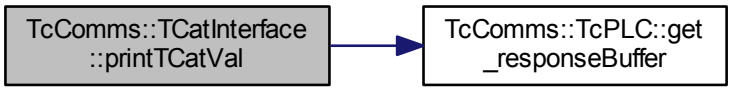
This is a function for printing the variable name and value of a record. Depending on the variable type, the readout from the ADS server is cast into the proper data type and printed to the output file fp.

Definition at line 441 of file tcComms.cpp.

References TcComms::TcPLC::get_responseBuffer(), TcComms::DataPar::length, parent, requestNum, requestOffs, tCatName, tCatSymbol, and tCatType.

```
442 {
448     fprintf(fp,"%15s: %15s         ",tCatName.c_str(), tCatType.c_str());
449
450     double              doublePLCVar;
451     float               floatPLCVar;
452     signed long int     sliPLCVar;
453     signed short int    ssiPLCVar;
454     signed char         charPLCVar;
455     char                chararrPLCVar[100];
456
457     TcPLC::buffer_ptr buf = parent->get_responseBuffer (
      requestNum);
458     if (!buf) return;
459     char* pTCatVal = buf.get() + requestOffs;
460     if (tCatType == "LREAL")
461     {
462         doublePLCVar    = *(double*)pTCatVal;
463         fprintf(fp,"%f",doublePLCVar);
464     }
465     else if (tCatType == "REAL")
466     {
467         floatPLCVar     = *(float*)pTCatVal;
468         fprintf(fp,"%f",floatPLCVar);
469     }
470     else if (tCatType == "DWORD" || tCatType == "DINT" ||
      tCatType == "UDINT")
471     {
472         sliPLCVar       = *(signed long int*)pTCatVal;
473         fprintf(fp,"%d",sliPLCVar);
474     }
475     else if (tCatType == "INT" || tCatType == "WORD" || tCatType == "ENUM" ||
      tCatType == "UINT")
476     {
477         ssiPLCVar       = *(signed short int*)pTCatVal;
478         fprintf(fp,"%d",ssiPLCVar);
479     }
480     else if (tCatType == "BOOL" || tCatType == "BYTE" ||
      tCatType == "SINT" || tCatType == "USINT")
481     {
482         charPLCVar      = *(signed char*)pTCatVal;
483         fprintf(fp,"%d",charPLCVar);
484     }
485     else if (tCatType.substr(0,6) == "STRING")
486     {
487         strncpy(chararrPLCVar, (char*)pTCatVal, tCatSymbol.length);
488         fprintf(fp,"%s",chararrPLCVar);
489     }
490     else
491     {
492         fprintf(fp,"INVALID!!!");
493     }
494
495     fprintf(fp,"\n");
496     //fprintf(fp,"      request %i       offset %i          \n", requestNum, tCatSymbol.indexOffset); //
       Use this to print additional information
497
498     if (debug)
499     {
500     }
501 }
```

Here is the call graph for this function:



The documentation for this class was generated from the following files:

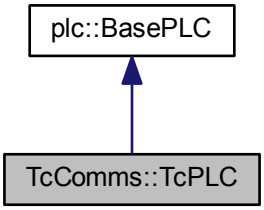
- tcComms.h
- tcComms.cpp

8.68 TcComms::TcPLC Class Reference

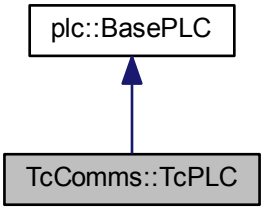
TwinCAT PLC.

```
#include <tcComms.h>
```

Inheritance diagram for TcComms::TcPLC:



Collaboration diagram for TcComms::TcPLC:



Public Types

- typedef char **buffer_type**
- typedef std::shared_ptr < buffer_type > **buffer_ptr**

Public Member Functions

- TcPLC (std::string tpyPath)

  Constructor.
- ~TcPLC ()

  Deconstructor.
- AmsAddr get_addr ()

  Get AMS netID of TwinCAT system and port number for this PLC.
- bool set_addr (std::stringcase netid, int port)
- long get_nReadPort ()

  Get read port number.
- long get_nWritePort ()

  Get write port number.
- int get_nRTS ()

  Get run-time system number.
- void set_nRTS (int RTS)

  Set run-time system number.
- void set_read_scanner_multiple (int mult)

  Set slowdown multiple for EPICS read.
- buffer_ptr get_responseBuffer (size_t idx)

  Get pointer to the beginning of a read request response buffer.
- bool optimizeRequests ()
- virtual bool start ()

  Starts the appropriate scanners.
- virtual void read_scanner ()

  Makes read requests to ADS, makes PlcWrite on all data values.
- virtual void write_scanner ()

  Collects records to be written to TCat, makes write request.
- virtual unsigned short checkState ()

  Sends request to ADS for PLC status.
- virtual void printAllRecords ()

  Prints symbol information for entire list of symbols to console.

Protected Member Functions

- long openPort ()

  Opens a new ADS communication port.
- void closePort (long nPort)

Protected Attributes

- AmsAddr addr

  AMS netID of TwinCAT system and port number for this PLC.
- int nRTS

  Run-time system number.
- std::string pathTpy

  The path of the tpy file.
- int nRequest

  Number of read request groups.
- std::vector< DataPar > adsGroupReadRequestVector

  Vector of index group, index offset, size for read requests.
- std::vector< buffer_ptr > adsResponseBufferVector

  Vector of buffers for each read request group.
- int scanRateMultiple

  Slowdown multiple for EPICS read.
- long nReadPort

  Port number for ADS read connection.
- long nWritePort

  Port number for ADS write connection.

8.68.1 Detailed Description

TwinCAT PLC.

Class for a connection to a TwinCAT PLC This class is derived from a BasePLC object, and specializes in managing records that contain a plc interface for TCat. This class is initialized using a .tpy file, from which it will obtain the AMS address information for connecting with TCat through ADS.

Reading and writing from/to ADS will be managed by this class, with read requests being grouped by continuous memory region in TCat to optimize read scanning for speed. Write requests are made using an ADS sum request.

There is also an option to send a request to ADS to check the status of both the PLC device and also the ADS connection.

Definition at line 268 of file tcComms.h.

8.68.2 Member Function Documentation

8.68.2.1 void TcComms::TcPLC::closePort (long *nPort*) [protected]

Closes an ADS communication port

Parameters

Number	of port to close

Definition at line 81 of file tcComms.cpp.

```
82 {
83     AdsPortCloseEx(nPort);
84 }
```

8.68.2.2 bool TcComms::TcPLC::optimizeRequests ( )

Sorts read channels into request groups. Will make a new request group for channels not in continuous memory region in TCat. Will create buffers of appropriate size for each read request, and let each TCat record know where in the read response buffer the data for that symbol is.

Returns

true if successful

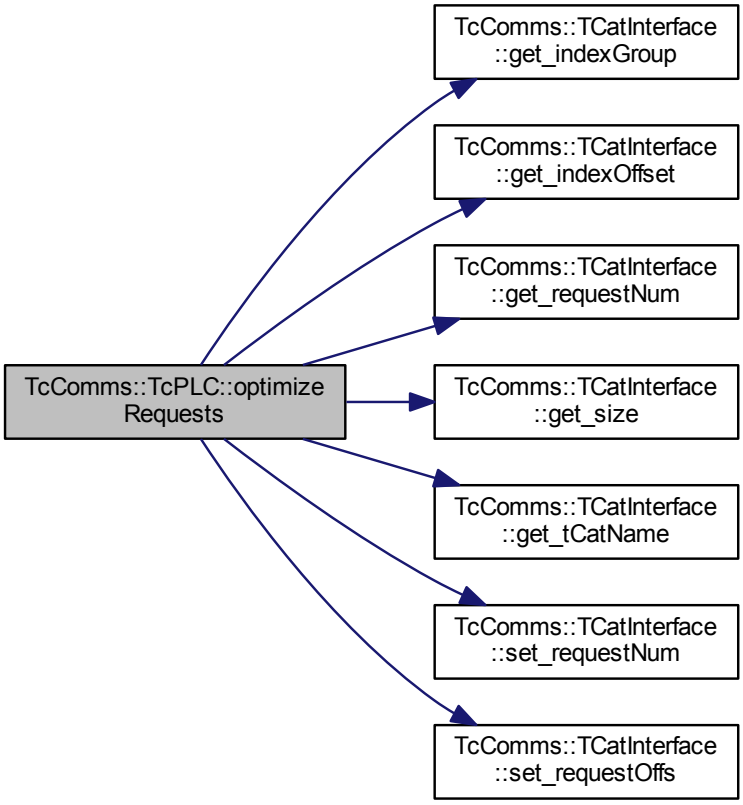
Definition at line 110 of file tcComms.cpp.

References TcComms::TCatInterface::get_indexGroup(), TcComms::TCatInterface::get_indexOffset(), TcComms::TCatInterface::get_requestNum(), TcComms::TCatInterface::get_size(), TcComms::TCatInterface::get_tCatName(), TcComms::DataPar::indexGroup, TcComms::DataPar::indexOffset, TcComms::DataPar::length, MAX_REL_GAP, MAX_REQ_SIZE, MAX_SINGLE_GAP_SIZE, MIN_REL_GAP_SIZE, TcComms::TCatInterface::set_requestNum(), and TcComms::TCatInterface::set_requestOffs().

```
111 {
112     // TODO: THIS FUNCTION NEEDS A NEW NAME
113     if (debug) printf("Forming requests...\n");
114
115     // Copy records into a list for sorting
116     std::list<BaseRecordPtr> recordList;
117     for (auto it = records.begin(); it!= records.end(); ++it) {
118         recordList.push_back( it->second);
119     }
120
121     // Sort record list by group and offset
122     recordList.sort( compByOffset );
123
124     bool gap;
125     int nextOffs;
126     auto it = recordList.begin();
127     TCatInterface* rec = dynamic_cast<TCatInterface*>((*it).get()->get_plcInterface());
128     if (!rec) return false;
129     DataPar request;
130     nextOffs = rec->get_indexOffset();
131     request.indexGroup = rec->get_indexGroup();
132     request.indexOffset = rec->get_indexOffset();
133     request.length = 0;
134     int totalGap = 0;
135     double relGap;
136     int nextLength = 0;
137     int nextGap;
138
139     // Iterate through the record list
140     for (it = recordList.begin(); it != recordList.end(); ++it)
141     {
142         rec = dynamic_cast<TCatInterface*>((*it).get()->get_plcInterface());
143         if (!rec) continue;
144         if (tcdebug) printf("Processing record: %s\n", rec->get_tCatName().c_str());
145
146         long recGroup = rec->get_indexGroup();
147         long recOffset = rec->get_indexOffset();
148         long recSize = rec->get_size();
149
150         nextGap = recOffset - nextOffs;
151         totalGap += nextGap;
152         nextLength = request.length + recSize + nextGap;
153         relGap = ((double) totalGap) / nextLength;
154
155         gap = (recOffset - nextOffs > MAX_SINGLE_GAP_SIZE || (totalGap >
    MIN_REL_GAP_SIZE && relGap > MAX_REL_GAP)) || recGroup != request.indexGroup;
156
157         // Make new request if offset gap > MAX_GAP_SIZE or if new index group or if request size too big
158         if (request.length + recSize > MAX_REQ_SIZE || gap)
159         {
160             if (debug) printf("Moving to next request... Gap size is %d\n",recOffset - nextOffs);
161             nRequest++;
162             adsGroupReadRequestVector.push_back(request);
163             request.indexGroup = recGroup;
164             request.indexOffset = recOffset;
165             request.length = recSize;
166             totalGap = 0;
167         }
168         else
169         {
170             request.length = nextLength;
171         }
```

```
        rec->set_requestNum(nRequest);
        nextOffs = request.indexOffset + request.length; // what the next offset should be for continuous
    }
    // Flush out last request
    adsGroupReadRequestVector.push_back(request);
    if (tcdebug) printf("length: %d, nextOffs: %d, totalGap: %d\n", request.length, nextOffs, totalGap);

    // Make response buffer
    if (debug) printf("Making buffer...\n");
    for (auto i : adsGroupReadRequestVector)
    {
        buffer_type* buffer = new (nothrow) buffer_type [i.length + 4];
        adsResponseBufferVector.push_back(buffer_ptr(buffer));
    }

    // Set offset into request buffer for each record
    int reqNum;
    int recOffs;
    int reqOffs;
    for (it = recordList.begin(); it != recordList.end(); ++it)
    {
        rec = dynamic_cast<TCatInterface*>((*it).get()->get_plcInterface());
        if (!rec) continue;
        reqNum = rec->get_requestNum();
        recOffs = rec->get_indexOffset();
        reqOffs = adsGroupReadRequestVector[reqNum].indexOffset;
        rec->set_requestOffs (recOffs - reqOffs);

        if (tcdebug) printf("Record %s linked to ADS response buffer.\n",rec->get_tCatName().c_str());
    }

    return true;
}
```

Here is the call graph for this function:



8.68.2.3 bool TcComms::TcPLC::set_addr (std::stringcase *netid*, int *port*)

Set AMS address

Returns

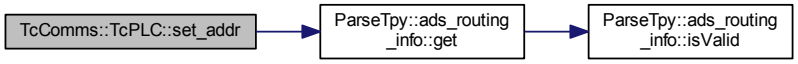
true if successful

Definition at line 60 of file tcComms.cpp.

References ParseTpy::ads_routing_info::get().

```
61 {
62     ParseTpy::ads_routing_info ads (netIdStr, port);
63     if (!ads.get (addr.netId.b[0], addr.netId.b[1], addr.netId.b[2],
64         addr.netId.b[3], addr.netId.b[4], addr.netId.b[5])) {
65         return false;
66     }
67     addr.port = port;
68     set_name (ads.get());
69     return true;
70 }
```

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- tcComms.h
- tcComms.cpp

8.69 TcComms::tcProcWrite Class Reference

TwinCAT process write requests.

`#include <tcComms.h>`

Public Member Functions

- tcProcWrite (const AmsAddr &a, long amsport, size_t mrec=1000)

  Default constructor.
- ~tcProcWrite ()

  Destructor: will porcess the TCat writes.
- tcProcWrite (tcProcWrite &&tp)

  Move constructor.
- void operator() (plc::BaseRecord *prec)

  Process on record.
- void * read_ptr (int sz)
- bool add (long igroup, long ioffs, long sz)

Protected Member Functions

- bool check_alloc (int extra=0)

  Checks if we have enough memory allocated.
- void tcwrite ()

  writes the current header/data to TCat
- tcProcWrite & operator= (tcProcWrite &&)

  Move operator.

Protected Attributes

- AmsAddr addr

  AMS address.
- long port

  Port to be used to write to TCat.
- char * ptr

  Pointer to header to be written.
- char * data

  Pointer to data to be written.
- size_t maxrec

  Maximum number of individual requests.
- size_t size

  Size of data.
- size_t alloc

  Size of allocated header/data array.
- size_t count

  Current number of individual requests.
- std::vector< tcProcWrite > req

8.69.1 Detailed Description

TwinCAT process write requests.

Class for collecting and processing write requests This class iterates through the entire record list on the PLC and collects those records whose data value has a dirty flag set on the plc side. These records are then sent as a group to ADS.

In order to not overload the ADS server, a maximum number of symbols per request is defined, and should not be > 2000.

Definition at line 189 of file tcComms.h.

8.69.2 Member Function Documentation

8.69.2.1 bool TcComms::tcProcWrite::add (long *igroup*, long *ioffs*, long *sz*)

Add header info

Parameters

igroup	iGroup number for tc write
ioffs	iOffset number for tc write
sz	Size of data to be written

Returns

true if succesful

Definition at line 304 of file tcComms.cpp.

```
{
    if (count == maxrec) {
        // Just queue it up!
        req.push_back (std::move (*this));
    }
    if (!check_alloc (0)) {
        return false;
    }
    char* p = ptr + count * 3 * sizeof (long);
    memcpy (p, &igroup, sizeof (long));
    memcpy (p + sizeof (long), &ioffs, sizeof (long));
    memcpy (p + 2 * sizeof (long), &sz, sizeof (long));
    ++count;
    return true;
}
```

8.69.2.2 void * TcComms::tcProcWrite::read_ptr (int *sz*)

Get a pointer to read the value in

Parameters

sz	Requested size

Definition at line 292 of file tcComms.cpp.

```
{
    if (!check_alloc (sz)) {
        return nullptr;
    }
    char* ret = data + size;
    size += sz;
    return ret;
}
```

8.69.3 Member Data Documentation

8.69.3.1 std::vector<tcProcWrite> TcComms::tcProcWrite::req [protected]

Queued TCat requests (each would have reached maxrec individual requests)

Definition at line 234 of file tcComms.h.

The documentation for this class was generated from the following files:

- tcComms.h
- tcComms.cpp

8.70 DevTc::tcRegisterTolocShell Class Reference

```
#include <drvTc.h>
```

8.70.1 Detailed Description

Register TC commands to IOC shell This class registers the callback functions for the TC IOC commands

Definition at line 27 of file drvTc.h.

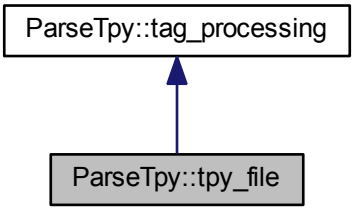
The documentation for this class was generated from the following files:

- drvTc.h
- drvTc.cpp

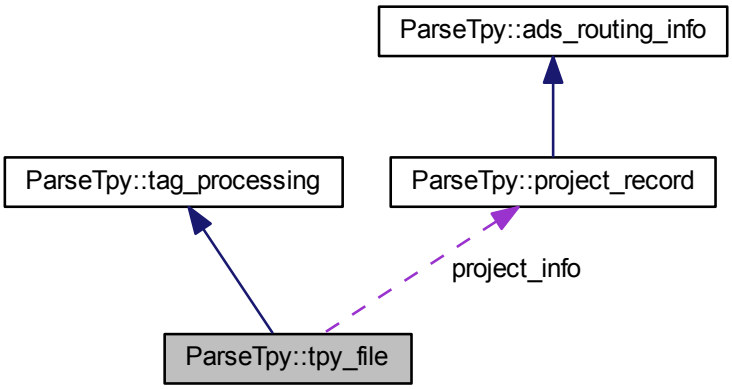
8.71 ParseTpy::tpy_file Class Reference

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::tpy_file:



Collaboration diagram for ParseTpy::tpy_file:



Public Member Functions

- tpy_file ()

Default constructor.

- tpy_file (FILE *inp)

  Constructor.
- bool parse (FILE *inp)

  Parse a file.
- bool parse (const char *p, int len)

  Parse a memory region.
- const symbol_list & get_symbols () const

  Return list of symbols.
- const type_map & get_types () const

  Return list of types.
- const project_record & get_project_info () const

  Return project information.
- template<class Function >
  int process_symbols (Function &process, const std::stringcase &prefix=std::stringcase()) const

  Process the type tree of a symbol.
- template<class Function >
  int process_type_tree (const symbol_record &symbol, Function &process, const std::stringcase &prefix=std::stringcase()) const

  Process the type tree of a symbol.
- template<class Function >
  int process_type_tree (const type_record &typ, opc_list defopc, memory_location loc, Function &process, std::stringcase varname=std::stringcase(), int level=0) const

  Process the type tree of a type.
- template<class Function >
  int process_type_tree (const std::stringcase &typ, unsigned int id, opc_list defopc, memory_location loc, Function &process, std::stringcase varname=std::stringcase(), int level=0) const

  Process the type tree of a type.

Protected Member Functions

- void parse_finish ()

  finish up the parsing
- template<class Function >
  int process_array (const type_record &typ, dimensions dim, opc_list defopc, memory_location loc, Function &process, const std::stringcase &varname, int level) const

  Process the type tree of a type.

Protected Attributes

- project_record project_info

  Project information.
- symbol_list sym_list

  List of symbols.
- type_map type_list

  List of types.

8.71.1 Detailed Description

This class holds the structure of a tpy file

Definition at line 617 of file ParseTpy.h.

8.71.2 Member Function Documentation

8.71.2.1 void ParseTpy::tpy_file::parse_finish ( ) [protected]

finish up the parsing

```
This function is called at the end of parsing.
```

Here we set the TC server name in the OPC variables for each symbol

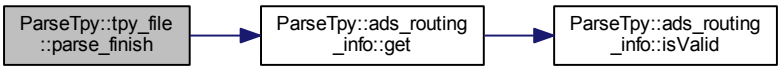
Definition at line 583 of file ParseTpy.cpp.

References ParseTpy::ads_routing_info::get(), project_info, and sym_list.

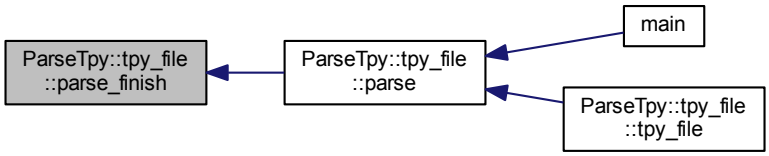
Referenced by parse().

```
584 {
585     std::stringcase tcname = project_info.get();
586     if (!tcname.empty()) {
587         for (auto sym = sym_list.begin(); sym != sym_list.end(); ++sym) {
588             sym->get_opc().add (property_el (OPC_PROP_PLCNAME, tcname));
589         }
590     }
591 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.71.2.2 template<class Function > int ParseTpy::tpy_file::process_array (const type_record & *typ*, dimensions *dim*, opc_list *defopc*, memory_location *loc*, Function & *process*, const std::stringcase & *varname*, int *level*) const [protected]

Process the type tree of a type.

```
Resolves the type information for an array. Calls the process
```

function for each index with an argument of type process_arg.

Parameters

typ	Name of type to resolve
id	Decoration or unique ID of type
dim	Dimensions of the array
defopc	Default list of OPC parameters
loc	Memory location of variable
process	Function class
varname	Name of variable of the specified type (default is "")
level	Recursive level (stops when reaching 100, default 0)

Returns

Number of processes variables

Definition at line 193 of file ParseTpyTemplate.h.

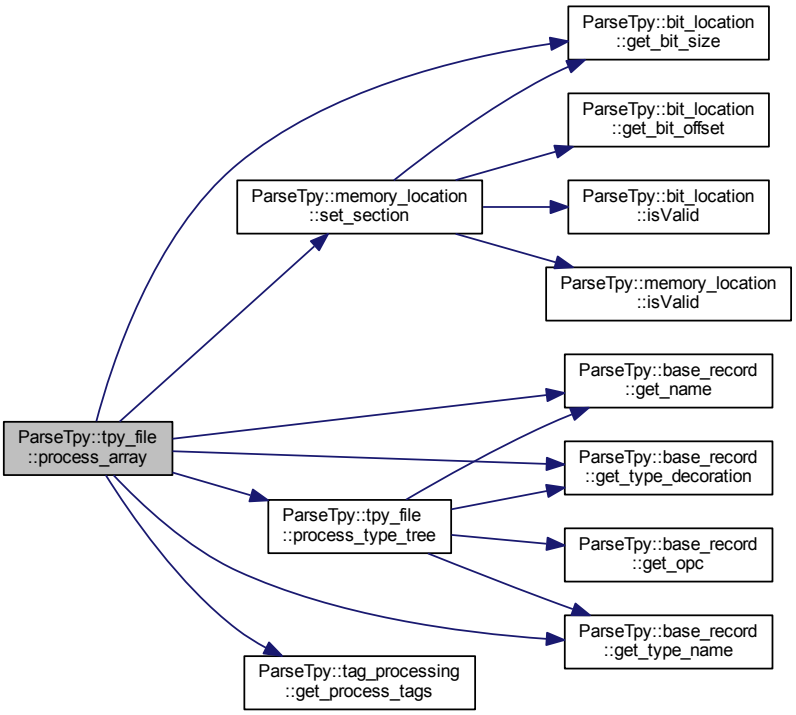
References ParseTpy::bit_location::get_bit_size(), ParseTpy::base_record::get_name(), ParseTpy::tag_processing::get_process_tags(), ParseTpy::base_record::get_type_decoration(), ParseTpy::base_record::get_type_name(), ParseTpy::process_all, ParseTpy::process_structured, process_type_tree(), ParseTpy::pt_binary, and ParseTpy::memory_location::set_section().

Referenced by process_type_tree().

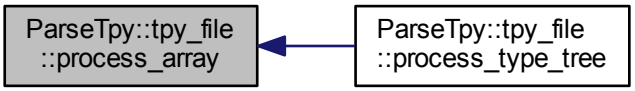
```
    {
        // This is an array where all dimensions have been processed
        if (dim.empty()) {
            return process_type_tree (typ.get_type_name(),
                typ.get_type_decoration(), defopc, loc, process,
                varname, level);
        }
        // If not process through the indices
        else {
            dimension d = dim.front();
            dim.pop_front();
            // invalid number of elements
            if (d.second < 0) {
                fprintf (stderr, "Array with negative element number for %s\n", varname.c_str());
                return 0;
            }
            // Call process for entire array (not an atomic type)
            process_arg arg (loc, varname, pt_binary, defopc, typ.get_name(),
                    typ.get_type_decoration(), false, &typ);
            int num = 0;
            if (get_process_tags() == process_structured ||
                get_process_tags() == process_all) {
                num += process (arg) ? 1 : 0;
            }

            // no elements
            if (d.second == 0) {
                // just ignore and continue
                return 0;
            }
            // valid number of elements
            else {
                // Check if we have a size which is a multiple of the array length
                if (typ.get_bit_size() % d.second != 0) {
                    fprintf (stderr, "Illegal array bit size for %s\n", varname.c_str());
                    return 0;
                }
            }
            // loop through first array dimension and call process_array for next dimension
            int el_bitsize = typ.get_bit_size() / d.second;
            for (int i = d.first; i < d.first + d.second; ++i) {
                memory_location el_loc = loc;
                bit_location el ((i - d.first) * el_bitsize, el_bitsize);
                if (!el_loc.set_section (el)) {
                    continue;
                }
                char buf[40];
                sprintf_s (buf, sizeof (buf), "[%i]", i);
                std::stringcase narr (varname);
                narr += buf;
                num += process_array (typ, dim, defopc, el_loc, process, narr, level);
            }
            return num;
```

```
        }
    }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.71.2.3 template<class Function > int ParseTpy::tpy_file::process_symbols (Function & *process*, const std::stringcase & *prefix* = std::stringcase ()) const

Process the type tree of a symbol.

```
Iterates over the symbol list and processes all specified tags.
```

Parameters

process	Function class
prefix	Prefix which is added to all variable names

Returns

Number of processes variables

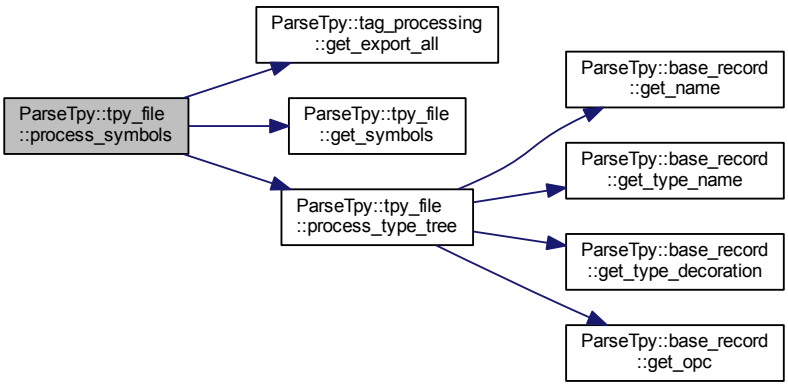
Definition at line 9 of file ParseTpyTemplate.h.

References ParseTpy::tag_processing::get_export_all(), get_symbols(), process_type_tree(), and ParseTpy::publish.

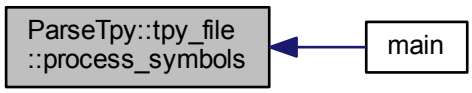
Referenced by main().

```
11      {
12          int num = 0;
13          for (symbol_list::const_iterator sym = get_symbols().begin();
14              sym != get_symbols().end(); ++sym, ++num) {
15              if (get_export_all() ||
16                  (sym->get_opc().get_opc_state() == publish)) {
17                  int ret = process_type_tree (*sym, process, prefix);
18                  num += ret;
19              }
20          }
21          return num;
22      }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.71.2.4 template<class Function > int ParseTpy::tpy_file::process_type_tree (const symbol_record & *symbol*, Function & *process*, const std::stringcase & *prefix* = std::stringcase ()) const

Process the type tree of a symbol.

```
Starts with a symbol and resolves the type information
```

recursevly until an atomic type (like INT) is found. Then, calls the process function with an argument of type process_arg. The function must return true if suffessful and false otherwise.

Parameters

symbol	Symbol to resolve
process	Function class
prefix	Prefix which is added to all variable names

Returns

Number of processes variables

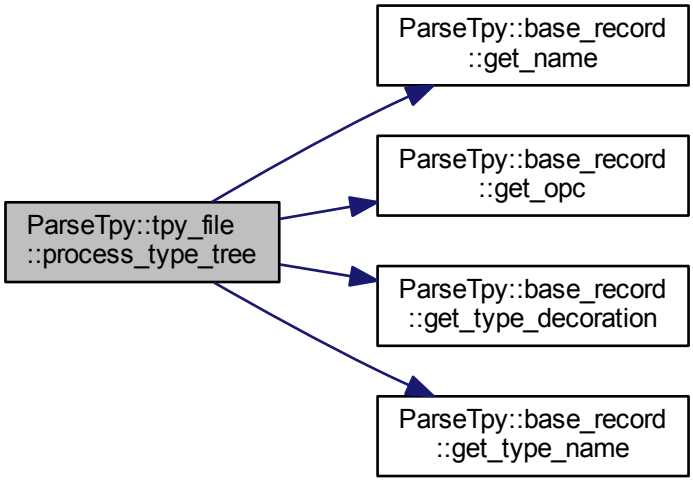
Definition at line 27 of file ParseTpyTemplate.h.

References ParseTpy::base_record::get_name(), ParseTpy::base_record::get_opc(), ParseTpy::base_record::get_type_decoration(), and ParseTpy::base_record::get_type_name().

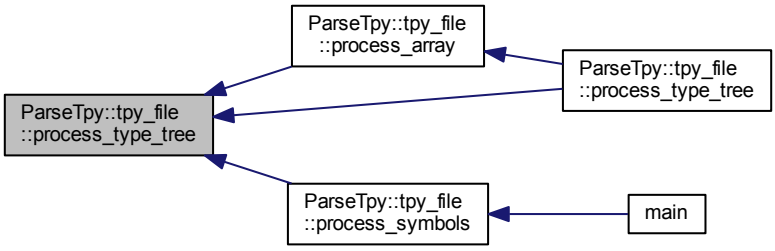
Referenced by process_array(), process_symbols(), and process_type_tree().

```
    {
        std::stringcase n = prefix + symbol.get_name();
//      if (!n.empty() && n[0] == '.') n.erase (0, 1);
        if (!n.empty()) {
            return process_type_tree (symbol.get_type_name(),
                symbol.get_type_decoration(),
                symbol.get_opc(), symbol, process, n, 0);
        }
        else {
            return 0;
        }
    }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.71.2.5 template<class Function > int ParseTpy::tpy_file::process_type_tree (const type_record & *typ*, opc_list *defopc*, memory_location *loc*, Function & *process*, std::stringcase *varname* = std::stringcase () , int *level* = 0) const

Process the type tree of a type.

```
Starts with a type and resolves the type information
```

recursevly until an atomic type (like BOOL) is found. Then, calls the process function with an argument of type process_arg.

Parameters

typ	Type to resolve
defopc	Default list of OPC parameters
loc	Memory location of variable
process	Function class
varname	Name of variable of the specified type (default is "")
level	Recursive level (stops when reaching 100, default 0)

Returns

Number of processes variables

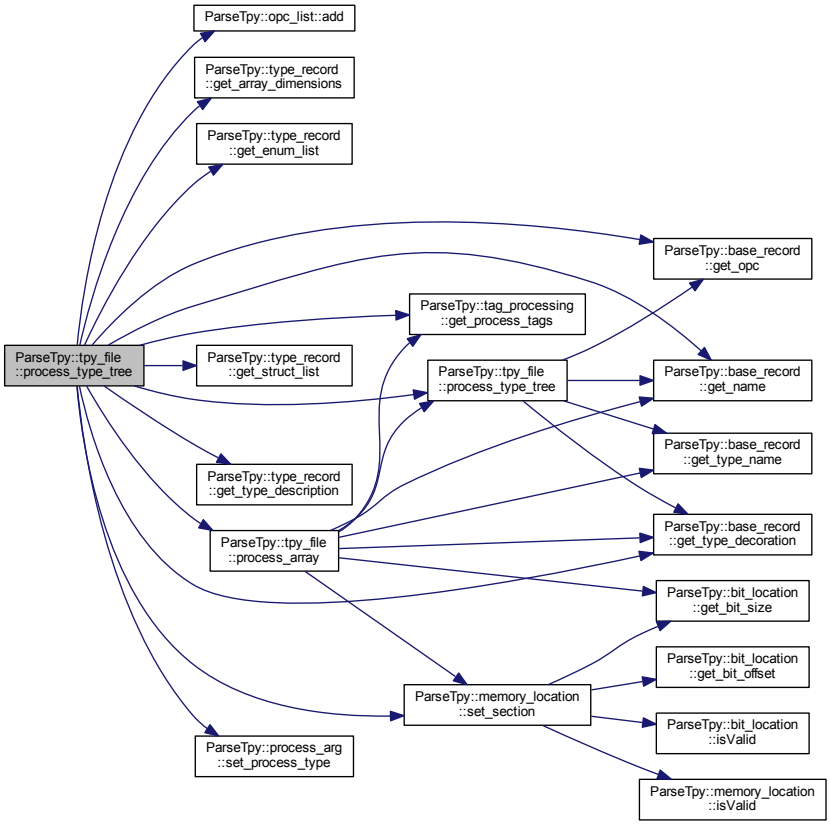
Definition at line 45 of file ParseTpyTemplate.h.

References ParseTpy::opc_list::add(), ParseTpy::arraytype, ParseTpy::enumtype, ParseTpy::functionblock, ParseTpy::type_record::get_array_dimensions(), ParseTpy::type_record::get_enum_list(), ParseTpy::base_record::get_name(), ParseTpy::base_record::get_opc(), ParseTpy::tag_processing::get_process_tags(), ParseTpy::type_record::get_struct_list(), ParseTpy::base_record::get_type_decoration(), ParseTpy::type_record::get_type_description(), ParseTpy::process_all, process_array(), ParseTpy::process_atomic, ParseTpy::process_structured, process_type_tree(), ParseTpy::pt_binary, ParseTpy::pt_enum, ParseTpy::pt_int, ParseTpy::pt_invalid, ParseTpy::process_arg::set_process_type(), ParseTpy::memory_location::set_section(), ParseTpy::simple, and ParseTpy::structtype.

```
48     {
49         // Check recursive level
50         if (level > 100) return 0;
51
52         // need to make sure that we don't add opc definitions when the array type was
53         // implicitly defined through a declaration of form A : ARRAY[n...m] OF ...
54         // multiple implicit array types of the same index range and base type will
55         // share a single type defintion and therefore share the opc definitions of
```

```
        // the first implicit one
        if ((typ.get_type_description() != arraytype) ||
            (typ.get_name().find ('[') == std::stringcase::npos)) {
                defopc.add (typ.get_opc());
        }
        switch (typ.get_type_description()) {
        case simple :
            // next level
            return process_type_tree (typ.get_name(),
                typ.get_type_decoration(), defopc, loc, process, varname, level);
        case arraytype :
            // process array and iterate over all subindices
            return process_array (typ, typ.get_array_dimensions(), defopc, loc,
                process, varname, level);
        case enumtype :
            if (get_process_tags() == process_atomic ||
                get_process_tags() == process_all) {
                // check if enum is contained with 0 to 15
                bool withinhex = true;
                for (enum_map::const_iterator e = typ.get_enum_list().begin();
                    e != typ.get_enum_list().end(); ++e) {
                        if ((e->first < 0) || (e->first >= 16)) {
                            withinhex = false;
                            break;
                        }
                }
                process_arg arg (loc, varname, pt_invalid, defopc, typ.get_name(),
                    typ.get_type_decoration(), true, &typ);
                // if not, treat as int
                if (!withinhex) {
                    arg.set_process_type (pt_int);
                    return process (arg) ? 1 : 0;
                }
                // add opc property for enum values
                else {
                    for (enum_map::const_iterator e = typ.get_enum_list().begin();
                        e != typ.get_enum_list().end(); ++e) {
                            arg.get_opc().get_properties().insert (
                                property_el (8510 + e->first, e->second));
                    }
                    arg.set_process_type (pt_enum);
                    return process (arg) ? 1 : 0;
                }
            }
        case structtype :
        case functionblock :
            {
                // Call process for entire array (not an atomic type)
                int num = 0;
                if (get_process_tags() == process_structured ||
                    get_process_tags() == process_all) {
                    process_arg arg (loc, varname, pt_binary, defopc, typ.get_name(),
                        typ.get_type_decoration(), false, &typ);
                    num = process (arg) ? 1 : 0;
                }
                // iterate over all structure items
                for (item_list::const_iterator i = typ.get_struct_list().begin();
                    i !=  typ.get_struct_list().end(); ++i) {
                    // calculate sub element location
                    memory_location el_loc = loc;
                    if (!el_loc.set_section (*i)) {
                        continue;
                    }
                    // Add a dot to the variable name
                    std::stringcase n (varname);
                    n += ".";
                    n += i->get_name();
                    opc_list o (defopc);
                    o.add (i->get_opc());
                    num += process_type_tree (i->get_type_name(),
                        i->get_type_decoration(), o, el_loc, process, n, level + 1);
                }
                return num;
            }
        default :
            fprintf (stderr, "Unknown type for %s\n", varname.c_str());
            return 0;
        }
    }
```

Here is the call graph for this function:



8.71.2.6 template<class Function > int ParseTpy::tpy_file::process_type_tree (const std::stringcase & *typ*, unsigned int *id*, opc_list *defopc*, memory_location *loc*, Function & *process*, std::stringcase *varname* = std::stringcase () , int *level* = 0) const

Process the type tree of a type.

```
Starts with a type and resolves the type information
```

recursevly until an atomic type (like STRING) is found. Then, calls the process function with an argument of type process_arg.

Parameters

typ	Name of type to resolve
id	Decoration or unique ID of type
defopc	Default list of OPC parameters
loc	Memory location of variable
process	Function class
varname	Name of variable of the specified type (default is "")

level	Recursive level (stops when reaching 100, default 0)

Returns

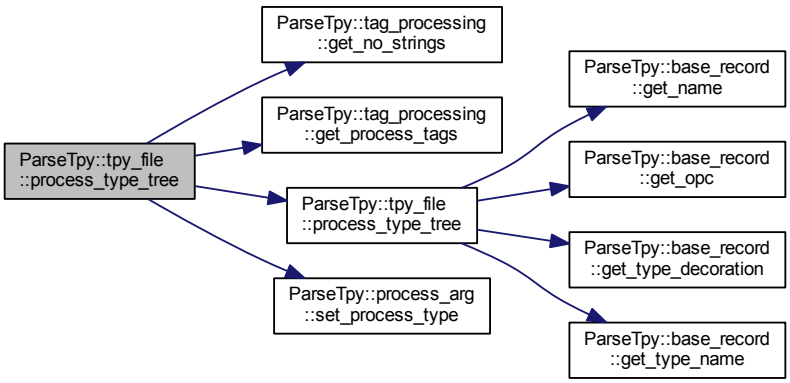
Number of processes variables

Definition at line 139 of file ParseTpyTemplate.h.

References ParseTpy::tag_processing::get_no_strings(), ParseTpy::tag_processing::get_process_tags(), ParseTpy::process_all, ParseTpy::process_atomic, process_type_tree(), ParseTpy::pt_bool, ParseTpy::pt_int, ParseTpy::pt_invalid, ParseTpy::pt_real, ParseTpy::pt_string, ParseTpy::process_arg::set_process_type(), and type_list.

```
142     {
143         type_map::const_iterator t;
144         process_arg arg (loc, varname, pt_invalid, defopc, typ);
145
146         if ((typ == "SINT")  || (typ == "INT")  || (typ == "DINT")  || (typ == "LINT")  ||
147             (typ == "USINT") || (typ == "UINT") || (typ == "UDINT") || (typ == "ULINT") ||
148             (typ == "BYTE")  || (typ == "WORD") || (typ == "DWORD") || (typ == "LWORD")) {
149             if (get_process_tags() == process_atomic ||
150                 get_process_tags() == process_all) {
151                 arg.set_process_type (pt_int);
152                 return process (arg) ? 1 : 0;
153             }
154             else return 0;
155         }
156         else if (typ == "REAL" || typ == "LREAL") {
157             if (get_process_tags() == process_atomic ||
158                 get_process_tags() == process_all) {
159                 arg.set_process_type (pt_real);
160                 return process (arg) ? 1 : 0;
161             }
162             else return 0;
163         }
164         else if (typ == "BOOL") {
165             if (get_process_tags() == process_atomic ||
166                 get_process_tags() == process_all) {
167                 arg.set_process_type (pt_bool);
168                 return process (arg) ? 1 : 0;
169             }
170             else return 0;
171         }
172         else if (typ.compare (0, 6, "STRING") == 0) {
173             if ((get_process_tags() == process_atomic ||
174                  get_process_tags() == process_all) &&
175                  !get_no_strings()) {
176                 arg.set_process_type (pt_string);
177                 return process (arg) ? 1 : 0;
178             }
179             else return 0;
180         }
181         else if ((t = type_list.find (id)) != type_list.end()) {
182             return process_type_tree (t->second, defopc, loc, process, varname, level);
183         }
184         else {
185             fprintf (stderr, "Unknown type for %s\n", varname.c_str());
186             return 0;
187         }
188     }
```

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- ParseTpy.h
- ParseTpy.cpp
- ParseTpyTemplate.h

8.72 DevTc::epics_record_traits< RecType >::traits_type Struct Reference

Epics record type.

```
#include <devTc.h>
```

Public Attributes

- double **val**

8.72.1 Detailed Description

template<epics_record_enum RecType>struct DevTc::epics_record_traits< RecType >::traits_type

Epics record type.

Definition at line 170 of file devTc.h.

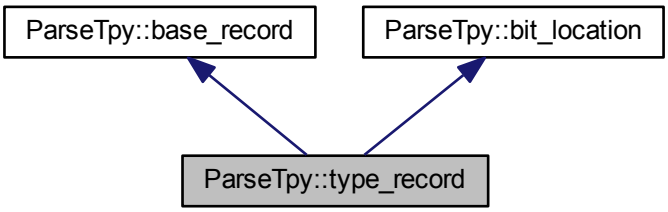
The documentation for this struct was generated from the following file:

- devTc.h

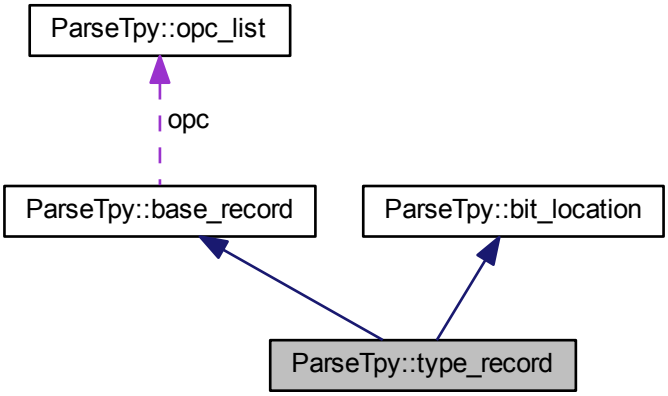
8.73 ParseTpy::type_record Class Reference

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::type_record:



Collaboration diagram for ParseTpy::type_record:



Public Member Functions

- type_record ()

  Default constructor.
- type_enum get_type_description () const

  get the data type
- void set_type_description (type_enum desc)

  Set decoration (type ID)
- unsigned int get_name_decoration () const

  Get decoration (type ID)
- void set_name_decoration (unsigned int id)

  Set decoration (type ID)
- const dimensions & get_array_dimensions () const

  Get array dimensions.
- dimensions & get_array_dimensions ()

Get array dimensions.

- const enum_map & get_enum_list () const

  Get enumerated list.
- enum_map & get_enum_list ()

  Get enumerated list.
- const item_list & get_struct_list () const

  Get structure list.
- item_list & get_struct_list ()

  Get structure list.

Protected Attributes

- type_enum type_desc

  Type description.
- unsigned int name_decoration

  decoration (type ID) of type name
- dimensions array_list

  table of dimensions
- enum_map enum_list

  map of enum id and name
- item_list **struct_subitems**

8.73.1 Detailed Description

This structure holds a type record

Definition at line 305 of file ParseTpy.h.

The documentation for this class was generated from the following file:

- ParseTpy.h

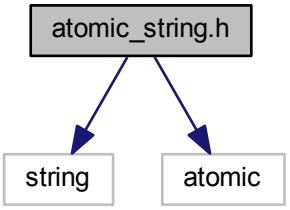
Chapter 9

File Documentation

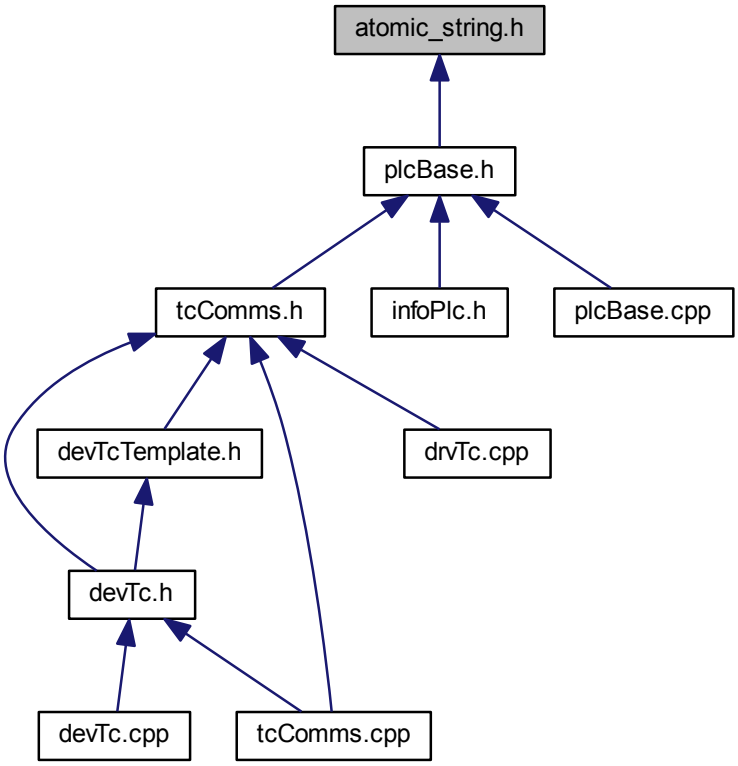
9.1 atomic_string.h File Reference

```
#include <string>
#include <atomic>
```

Include dependency graph for atomic_string.h:



This graph shows which files directly or indirectly include this file:



Classes

- class std::atomic_string< stringT >

  atomic strings
- class std::atomic< string >

  atomic<string>
- class std::atomic< wstring >

  atomic<wstring>

9.1.1 Detailed Description

Header which includes a class for implementing an atomic specialization for string.

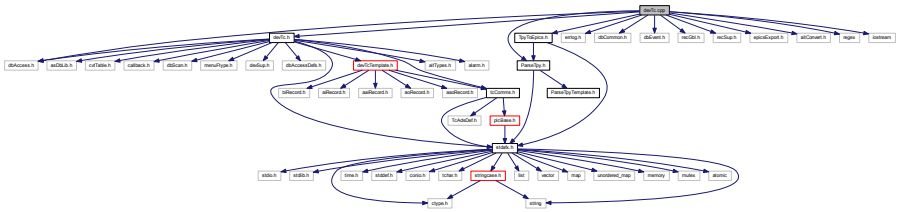
Definition in file atomic_string.h.

9.2 devTc.cpp File Reference

```
#include "devTc.h"
```

```
#include "ParseTpy.h"
#include "TpyToEpics.h"
#include "errlog.h"
#include "dbAccess.h"
#include "dbCommon.h"
#include "dbEvent.h"
#include "recGbl.h"
#include "recSup.h"
#include "epicsExport.h"
#include "aitConvert.h"
#include <regex>
#include <iostream>
```

Include dependency graph for devTc.cpp:



Namespaces

- DevTc

  Namespace for TCat device support.

Constant Groups

- DevTc

  Namespace for TCat device support.

Functions

- bool linkRecord (std::stringcase name, dbCommon *pEpicsRecord, BaseRecordPtr &pRecord)

  linkRecord
- **epicsExportAddress** (dset, aival_record_tc_dset)
- **epicsExportAddress** (dset, airval_record_tc_dset)
- **epicsExportAddress** (dset, aaival_record_tc_dset)
- **epicsExportAddress** (dset, bival_record_tc_dset)
- **epicsExportAddress** (dset, birval_record_tc_dset)
- **epicsExportAddress** (dset, longinval_record_tc_dset)
- **epicsExportAddress** (dset, mbbival_record_tc_dset)
- **epicsExportAddress** (dset, mbbirval_record_tc_dset)
- **epicsExportAddress** (dset, mbbiDirectval_record_tc_dset)
- **epicsExportAddress** (dset, mbbiDirectrval_record_tc_dset)
- **epicsExportAddress** (dset, stringinval_record_tc_dset)
- **epicsExportAddress** (dset, waveformval_record_tc_dset)
- **epicsExportAddress** (dset, aoval_record_tc_dset)
- **epicsExportAddress** (dset, aorval_record_tc_dset)
- **epicsExportAddress** (dset, aaoval_record_tc_dset)
- **epicsExportAddress** (dset, boval_record_tc_dset)

- **epicsExportAddress** (dset, borval_record_tc_dset)
- **epicsExportAddress** (dset, longoutval_record_tc_dset)
- **epicsExportAddress** (dset, mbboval_record_tc_dset)
- **epicsExportAddress** (dset, mbborval_record_tc_dset)
- **epicsExportAddress** (dset, mbboDirectval_record_tc_dset)
- **epicsExportAddress** (dset, mbboDirectrval_record_tc_dset)
- **epicsExportAddress** (dset, stringoutval_record_tc_dset)

9.2.1 Detailed Description

Methods for TwinCAT/ADS device support.

Definition in file devTc.cpp.

9.2.2 Function Documentation

9.2.2.1 bool linkRecord (std::stringcase *name*, dbCommon * *pEpicsRecord*, BaseRecordPtr & *pRecord*)

linkRecord

Initialization function that matches an EPICS record with an internal record entry

Parameters

name	Name of record
pEpicsRecord	Pointer to EPICS record
pRecord	Pointer to a base record

Returns

true if successful

Definition at line 51 of file devTc.cpp.

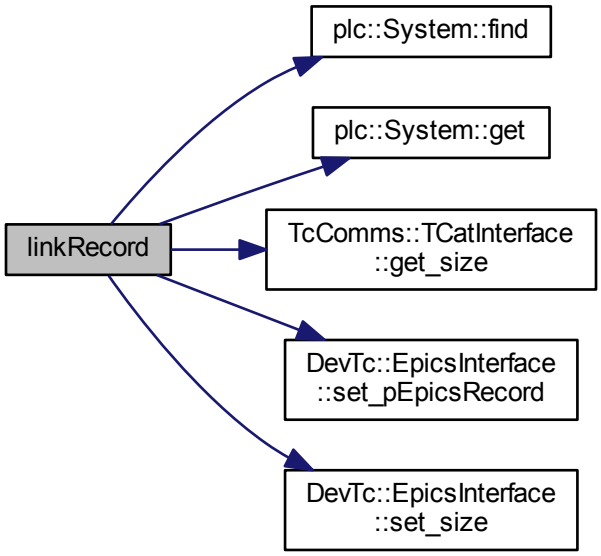
References plc::System::find(), plc::System::get(), TcComms::TCatInterface::get_size(), DevTc::EpicsInterface::set_pEpicsRecord(), and DevTc::EpicsInterface::set_size().

Referenced by DevTc::devTcDefIn< RecType >::init_read_record(), and DevTc::devTcDefOut< RecType >::init_write_record().

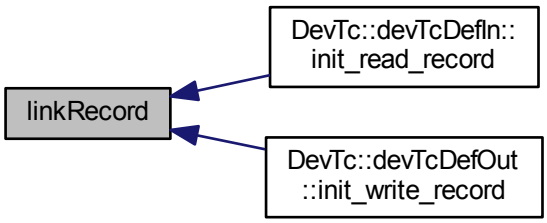
```
{
    static int inpError = 0;
    if (name.empty() ) {
        printf("Error in inp field for record %s.\n", pEpicsRecord->name);
        inpError++;
        return false;
    }
    std::regex rgx("((tc)://((\\b([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\\.?)+:(8[0-9][0-9]))/)(\\d{1,5})/(\\d{1,5}):(\\d{1,3})");

    std::smatch match;
    if (!std::regex_search(name, match, rgx)) {
        return false;
    }
    //for (auto m:match) std::cout << "submatch " << m << '\n';

    // Get PLC name from EPICS name string
    BasePLCPtr plcMatch = plc::System::get().find (match[1].str().c_str());
    if (!plcMatch.get()) {
        return false;
    }
    // Link record object to EPICS record
    pRecord = plcMatch->find (stringcase(name.c_str()));
    if (!pRecord.get()) {
        return false;
    }
    // Create EPICS interface
    DevTc::EpicsInterface* epics = new
```

```
    DevTc::EpicsInterface (*pRecord);

    // Let EPICS interface know about size of data
    TcComms::TCatInterface* tcat = (TcComms::TCatInterface*)
    pRecord->get_plcInterface();
    epics->set_size(tcat->get_size());

    // Link EPICS record to EPICS interface
    epics->set_pEpicsRecord(pEpicsRecord);
    // Link EPICS interface to record object
    pRecord->set_userInterface(epics);
    return true;

}
```

Here is the call graph for this function:



Here is the caller graph for this function:

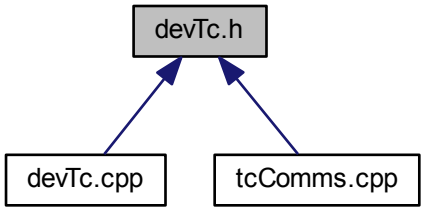


9.3 devTc.h File Reference

```
#include "stdafx.h"
#include "dbAccess.h"
#include "aitTypes.h"
#include "alarm.h"
#include "asDbLib.h"
#include "cvtTable.h"
#include "callback.h"
#include "dbScan.h"
#include "menuFtype.h"
#include "devSup.h"
#include "dbAccessDefs.h"
#include "tcComms.h"
#include "devTcTemplate.h"
```

Include dependency graph for devTc.h:

This graph shows which files directly or indirectly include this file:



Classes

- class DevTc::EpicsInterface

  epics interface class.
- struct DevTc::epics_record_traits< RecType >

  epics record traits.
- struct DevTc::epics_record_traits< RecType >::traits_type

  Epics record type.
- struct DevTc::devTcDefIo< RecType >

  device support record.
- struct DevTc::devTcDefIn< RecType >

  device support input record.
- struct DevTc::devTcDefOut< RecType >

device support output record.

- struct DevTc::devTcDefWaveformIn< RecType >

  device support waveform record.

Namespaces

- DevTc

  Namespace for TCat device support.

Constant Groups

- DevTc

  Namespace for TCat device support.

Enumerations

- enum DevTc::epics_record_enum {
  DevTc::aaival = 0, DevTc::aaoval, DevTc::aival, DevTc::aoval,
  DevTc::bival, DevTc::boval, DevTc::eventval, DevTc::histogramval,
  DevTc::longinval, DevTc::longoutval, DevTc::mbbival, DevTc::mbboval,
  DevTc::mbbiDirectval, DevTc::mbboDirectval, DevTc::stringinval, DevTc::stringoutval,
  DevTc::waveformval, DevTc::airval, DevTc::aorval, DevTc::birval,
  DevTc::borval, DevTc::mbbiDirectrval, DevTc::mbboDirectrval, DevTc::mbbirval,
  DevTc::mbborval, DevTc::epics_record_enumEnd, DevTc::invalidval = -1 }

  epics record type enum.

9.3.1 Detailed Description

Header which includes classes for TwinCAT/ADS device support.

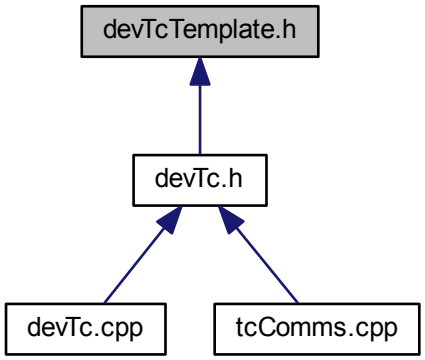
Definition in file devTc.h.

9.4 devTcTemplate.h File Reference

```
#include "tcComms.h"
#include "aiRecord.h"
#include "aaiRecord.h"
#include "aoRecord.h"
#include "aaoRecord.h"
#include "biRecord.h"
#include "boRecord.h"
#include "longinRecord.h"
#include "longoutRecord.h"
#include "stringinRecord.h"
#include "stringoutRecord.h"
#include "mbbiRecord.h"
#include "mbboRecord.h"
#include "mbbiDirectRecord.h"
#include "mbboDirectRecord.h"
#include "waveformRecord.h"
#include "eventRecord.h"
#include "histogramRecord.h"
```

Include dependency graph for devTcTemplate.h:

This graph shows which files directly or indirectly include this file:



Classes

- struct DevTc::epics_record_traits< aaival >

  Epics traits class specialization for aai record.
- struct DevTc::epics_record_traits< aaoval >

  Epics traits class specialization for aao record.
- struct DevTc::epics_record_traits< aival >

  Epics traits class specialization for ai record.
- struct DevTc::epics_record_traits< airval >

  Epics traits class specialization for ai raw record.
- struct DevTc::epics_record_traits< aoval >

  Epics traits class specialization for ao record.
- struct DevTc::epics_record_traits< aorval >

  Epics traits class specialization for ao raw record.
- struct DevTc::epics_record_traits< bival >

  Epics traits class specialization for bi record.
- struct DevTc::epics_record_traits< birval >

  Epics traits class specialization for bi raw record.
- struct DevTc::epics_record_traits< boval >

  Epics traits class specialization for bo record.
- struct DevTc::epics_record_traits< borval >

  Epics traits class specialization for bo raw record.
- struct DevTc::epics_record_traits< longinval >

  Epics traits class specialization for longin record.

- struct DevTc::epics_record_traits< longoutval >

  Epics traits class specialization for longout record.
- struct DevTc::epics_record_traits< mbbival >

  Epics traits class specialization for mbbi record.
- struct DevTc::epics_record_traits< mbbirval >

  Epics traits class specialization for mbbi raw record.
- struct DevTc::epics_record_traits< mbboval >

  Epics traits class specialization for mbbo record.
- struct DevTc::epics_record_traits< mbborval >

  Epics traits class specialization for mbbo raw record.
- struct DevTc::epics_record_traits< mbbiDirectval >

  Epics traits class specialization for mbbiDirect record.
- struct DevTc::epics_record_traits< mbbiDirectrval >

  Epics traits class specialization for mbbiDirect raw record.
- struct DevTc::epics_record_traits< mbboDirectval >

  Epics traits class specialization for mbboDirect record.
- struct DevTc::epics_record_traits< mbboDirectrval >

  Epics traits class specialization for mbboDirect raw record.
- struct DevTc::epics_record_traits< stringinval >

  Epics traits class specialization for stringin record.
- struct DevTc::epics_record_traits< stringoutval >

  Epics traits class specialization for stringout record.
- struct DevTc::epics_record_traits< waveformval >

  Epics traits class specialization for waveform record.
- struct DevTc::epics_record_traits< eventval >

  Epics traits class specialization for event record.
- struct DevTc::epics_record_traits< histogramval >

  Epics traits class specialization for histogram record.

Namespaces

- DevTc

  Namespace for TCat device support.

Constant Groups

- DevTc

  Namespace for TCat device support.

9.4.1 Detailed Description

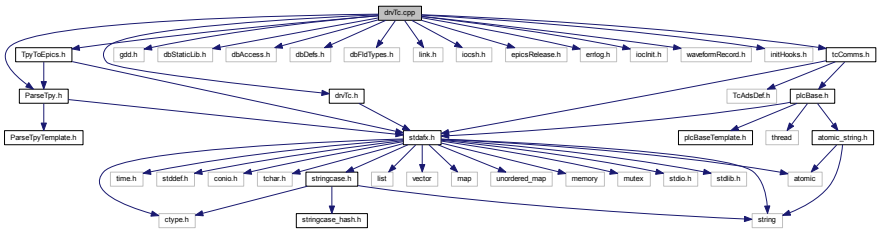
Header which includes templates for the device support functions for different EPICS record types.

Definition in file devTcTemplate.h.

9.5 drvTc.cpp File Reference

```
#include "drvTc.h"
#include "ParseTpy.h"
#include "TpyToEpics.h"
#include "gdd.h"
#include "dbStaticLib.h"
#include "dbAccess.h"
#include "dbDefs.h"
#include "dbFldTypes.h"
#include "link.h"
#include "iocsh.h"
#include "epicsRelease.h"
#include "errlog.h"
#include "iocInit.h"
#include "waveformRecord.h"
#include "initHooks.h"
#include "tcComms.h"
```

Include dependency graph for drvTc.cpp:



Classes

- class DevTc::epics_tc_db_processing

Namespaces

- DevTc

  Namespace for TCat device support.

Constant Groups

- DevTc

  Namespace for TCat device support.

Functions

- void **DevTc::tcLoadRecords** (const iocshArgBuf ∗args)
- void **DevTc::tcSetScanRate** (const iocshArgBuf ∗args)
- void **DevTc::tcPrintVals** (const iocshArgBuf ∗args)

Variables

- bool **dbg** = 1
- iocshFuncDef **DevTc::tcLoadRecordsFuncDef** = {"tcLoadRecords", 1, tcLoadRecordsArg}
- iocshFuncDef **DevTc::tcSetScanRateFuncDef** = {"tcSetScanRate", 2, tcSetScanRateArg}
- iocshFuncDef **DevTc::tcPrintValsFuncDef** = {"tcPrintVals", 1, tcPrintValsArg}

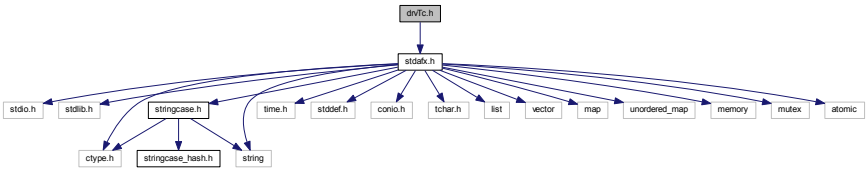
9.5.1 Detailed Description

This contains functions for driver support for EPICS. These routines are used during the IOC initialization, and allow the IOC to initialize the TwinCAT interface during EPICS initialization.

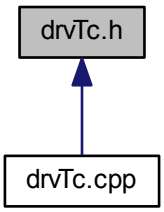
Definition in file drvTc.cpp.

9.6 drvTc.h File Reference

`#include "stdafx.h"`
Include dependency graph for drvTc.h:



This graph shows which files directly or indirectly include this file:



Classes

- class DevTc::tcRegisterToIocShell

Namespaces

- DevTc

  Namespace for TCat device support.

Constant Groups

- DevTc

  Namespace for TCat device support.

9.6.1 Detailed Description

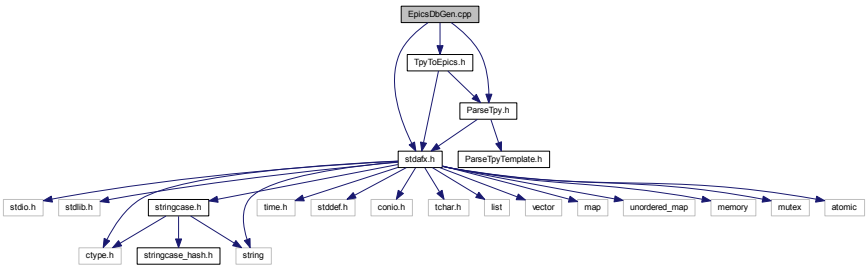
Header which includes classes for accessing TwinCAT/ADS from EPICS.

Definition in file drvTc.h.

9.7 EpicsDbGen.cpp File Reference

```
#include "stdafx.h"
#include "ParseTpy.h"
#include "TpyToEpics.h"
```

Include dependency graph for EpicsDbGen.cpp:



Functions

- int main (int argc, char ∗argv[])

9.7.1 Detailed Description

Source for the main program that generates an EPICs .db file

Definition in file EpicsDbGen.cpp.

9.7.2 Function Documentation

9.7.2.1 int main (int *argc*, char ∗ *argv[]*)

Main program

Definition at line 17 of file EpicsDbGen.cpp.

References EpicsTpy::device_support_opc_name, EpicsTpy::device_support_tc_name, EpicsTpy::split_io_support::flush(), EpicsTpy::split_io_support::get_processed_io(), EpicsTpy::split_io_support::get_processed_readonly(), EpicsTpy::split_io_support::get_processed_total(), EpicsTpy::ligo_std, EpicsTpy::listing_autoburt, EpicsTpy::listing_epics, EpicsTpy::listing_standard, EpicsTpy::lower_case, EpicsTpy::no_conversion, EpicsTpy::no_dot, ParseTpy::tpy_file::parse(), EpicsTpy::preserve_case, ParseTpy::process_all, ParseTpy::process_atomic,

ParseTpy::tpy_file::process_symbols(), EpicsTpy::epics_conversion::set_array_rule(), EpicsTpy::epics_conversion::set_case_rule(), EpicsTpy::epics_conversion::set_conversion_rule(), EpicsTpy::epics_db_processing::set_device_support(), EpicsTpy::epics_conversion::set_dot_rule(), ParseTpy::tag_processing::set_export_all(), EpicsTpy::epics_list_processing::set_listing(), ParseTpy::tag_processing::set_no_strings(), ParseTpy::tag_processing::set_process_tags(), EpicsTpy::epics_list_processing::set_verbose(), and EpicsTpy::upper_case.

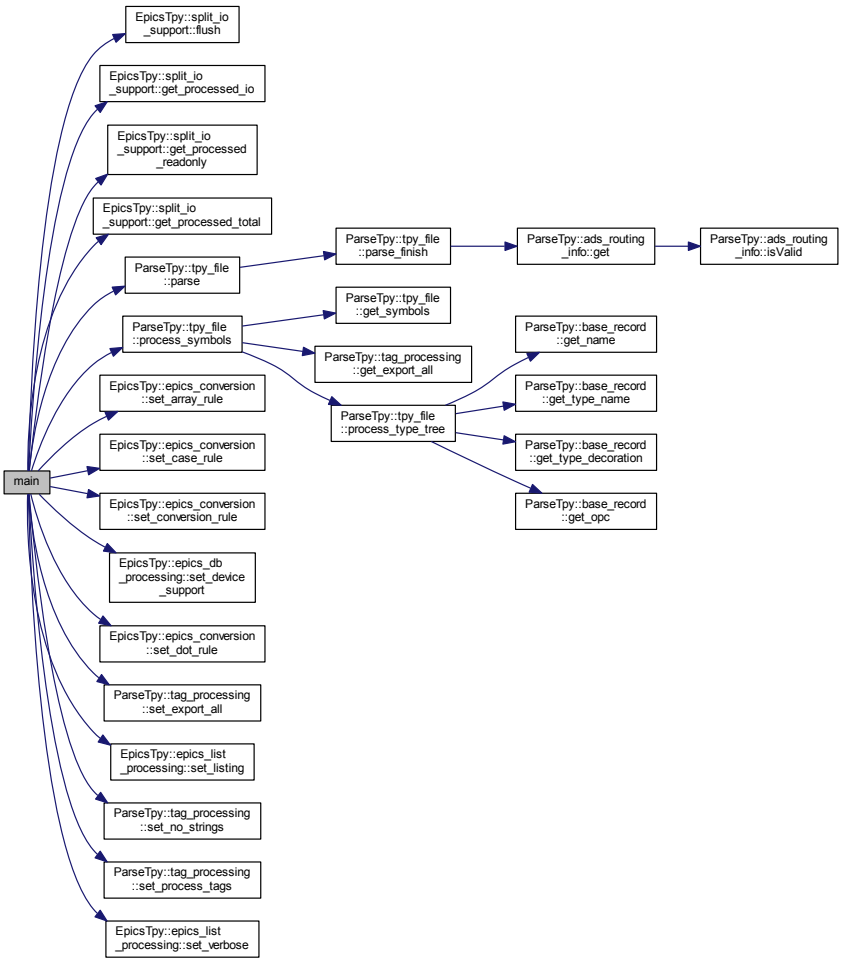
```
{
    tpy_file        tpyfile;
    stringcase        inpfilename;

    stringcase        outfilename;
    bool            split_io = false;
    int             split_n = 0;

    bool            listing = false;
    epics_list_processing  listproc;
    epics_db_processing      dbproc;

    int             help = 0;

    // default conversion rules
    *(epics_conversion*)(&listproc) =
        epics_conversion (ligo_std, preserve_case, false, false);
    *(epics_conversion*)(&dbproc) =
        epics_conversion (ligo_std, preserve_case, false, false);

    // command line parsing
    for (int i = 1; i < argc; ++i) {
        stringcase arg (argv[i] ? argv[i] : 0);
        stringcase next (i + 1 < argc && argv[i+1] ? argv[i+1] : "");
        // specify input file
        if ((arg == "-i" || arg == "/i") && i + 1 < argc) {
            inpfilename = next;
            i += 1;
        }
        // specify output file
        else if ((arg == "-o" || arg == "/o") && i + 1 < argc) {
            outfilename = next;
            i += 1;
        }
        // ask for help
        else if (arg == "-h" || arg == "/h" ) {
            help = 1;
        }

        // generate a standard listing
        else if (arg == "-l" || arg == "/l") {
            listing = true;
            listproc.set_listing (listing_standard);
            listproc.set_verbose (false);
            tpyfile.set_process_tags (process_all);
        }
        // generate a standard long listing
        else if (arg == "-ll" || arg == "/ll") {
            listing = true;
            listproc.set_listing (listing_standard);
            listproc.set_verbose (true);
            tpyfile.set_process_tags (process_all);
        }
        // generate a standard listing for atomic variables
        else if (arg == "-la" || arg == "/la") {
            listing = true;
            listproc.set_listing (listing_standard);
            listproc.set_verbose (false);
            tpyfile.set_process_tags (process_atomic);
        }
        // generate a standard long listing for atomic variables
        else if (arg == "-lla" || arg == "/lla") {
            listing = true;
            listproc.set_listing (listing_standard);
            listproc.set_verbose (true);
            tpyfile.set_process_tags (process_atomic);
        }
        // generate an EPICS name listing
        else if (arg == "-le" || arg == "/le") {
            listing = true;
            listproc.set_listing (listing_epics);
            listproc.set_verbose (false);
            tpyfile.set_process_tags (process_atomic);
        }
        // generate an EPICS name long listing
        else if (arg == "-lle" || arg == "/lle") {
            listing = true;
            listproc.set_listing (listing_epics);
            listproc.set_verbose (true);
```

```
            tpyfile.set_process_tags (process_atomic);
        }
        // generate a burt save/restore listing
        else if (arg == "-lb" || arg == "/lb" ||
                 arg == "-llb" || arg == "/llb") {
            listing = true;
            listproc.set_listing (listing_autoburt);
            listproc.set_verbose (false);
            tpyfile.set_process_tags (process_atomic);
        }

        // export only opc variables (default)
        else if (arg == "-eo" || arg == "/eo" ) {
            tpyfile.set_export_all (false);
        }
        // export all
        else if (arg == "-ea" || arg == "/ea" ) {
            tpyfile.set_export_all (true);
        }
        // Exclude string types (default)
        else if (arg == "-ns" || arg == "/ns") {
            tpyfile.set_no_strings (true);
        }
        // Include string types
        else if (arg == "-ys" || arg == "/ys") {
            tpyfile.set_no_strings (false);
        }
        // Replace array brackets with '_' (default)
        else if (arg == "-ni" || arg == "/ni") {
            listproc.set_array_rule (true);
            dbproc.set_array_rule (true);
        }
        // Keep array brackets
        else if (arg == "-yi" || arg == "/yi") {
            listproc.set_array_rule (false);
            dbproc.set_array_rule (false);
        }
        // Eliminate leading dot
        else if (arg == "-nd" || arg == "/nd") {
            listproc.set_dot_rule (true);
            dbproc.set_dot_rule (true);
        }
        // Keep leading do (default)
        else if (arg == "-yd" || arg == "/yd") {
            listproc.set_dot_rule (false);
            dbproc.set_dot_rule (false);
        }
        // Use TiwnCAT names
        else if (arg == "-no" || arg == "/no") {
            dbproc.set_device_support (device_support_tc_name);
        }
        // Used OPC names (default)
        else if (arg == "-yo" || arg == "/yo") {
            dbproc.set_device_support (device_support_opc_name);
        }

        // apply LIGO name conversion rules (default)
        else if (arg == "-rl" || arg == "/rl") {
            listproc.set_conversion_rule (ligo_std);
            dbproc.set_conversion_rule (ligo_std);
        }
        // apply no name conversion rules
        else if (arg == "-rn" || arg == "/rn") {
            listproc.set_conversion_rule (no_conversion);
            dbproc.set_conversion_rule (no_conversion);
        }
        // apply no name conversion rules
        else if (arg == "-rd" || arg == "/rd") {
            listproc.set_conversion_rule (no_dot);
            dbproc.set_conversion_rule (no_dot);
        }
        // preserve case during name conversion (default)
        else if (arg == "-cp" || arg == "/cp") {
            listproc.set_case_rule (preserve_case);
            dbproc.set_case_rule (preserve_case);
        }
        // force lower case during name conversion
        else if (arg == "-cu" || arg == "/cu") {
            listproc.set_case_rule (upper_case);
            dbproc.set_case_rule (upper_case);
        }
        // force upper case during name conversion
        else if (arg == "-cl" || arg == "/cl") {
            listproc.set_case_rule (lower_case);
            dbproc.set_case_rule (lower_case);
        }

```

```
        // split output files into readonly and in/out variables
        else if (arg == "-sio" || arg == "/sio" ||
                 arg == "-ysio" || arg == "/ysio") {
            split_io = true;
        }
        // Don't split output into readonly and in/out variables (default)
        else if (arg == "-nsio" || arg == "/nsio") {
            split_io = false;
        }
        // Specify a maximum number of variables per file (0/no default)
        else if ((arg == "-sn" || arg == "/sn") && i + 1 < argc) {
            split_n = atoi (next.c_str());
            if (split_n <= 0) split_n = 0;
            else if (split_n < 10) split_n = 10;
            i += 1;
        }
        else {
            help = 2;
        }
    }
    if (help) {
        printf ("Usage: EpicsDbGen ['options'] -i 'input' -o 'output'\n"
            "       Generates an EPICS database from a TwinCAT tpy file.\n"
            "       -ea exports all variables regardless of their opc setting\n"
            "       -l[l][a|e|b] generate an [extended] [atomic|epics|burt] channel listing\n"
            "       -r[n|d] [no|dot] conversion rule for EPICS names\n"
            "       -c[u|l] force upper/lower case for EPICS names\n"
            "       -nd eliminates leading dot\n"
            "       -no uses TwinCAT name in INPUT/OUTPUT fields instead of OPC\n"
            "       -ni replaces array brackets with underscore\n"
            "       -ns ignores channels of type string\n"
            "       -sio splits database into input only and input/ouput recrods\n"
            "       -sn 'num' splits database into files with no more than num records\n"
            "       -i 'input' input file name (stdin when omitted)\n"
            "       -o 'output' output database file (stdout when omitted)\n");
        if (help == 2) return 1;
        else return 0;
    }

    // open input file
    FILE* inpf = stdin;
    if (!inpfilename.empty()) {
        inpf = fopen (inpfilename.c_str(), "r");
        if (!inpf) {
            fprintf (stderr, "Failed to open input %s.\n", inpfilename.c_str());
            return 1;
        }
    }
    fprintf (stderr, "\nInput from %s\n",
        inpf == stdin ? "stdin" : inpfilename.c_str());

    // set up output
    split_io_support iosupp (outfilename, split_io, split_n);
    if (!iosupp) {
        fprintf (stderr, "Failed to open output %s.\n", outfilename.c_str());
        return 1;
    }
    (split_io_support&)(listproc) = iosupp;
    (split_io_support&)(dbproc) = iosupp;

    // print status of output
    fprintf (stderr, "Output to %s\n",
        iosupp.get_file() == stdout ? "stdout" : outfilename.c_str());
    fprintf (stderr, "Arguments are");
    // print arguments
    for (int i = 1; i < argc; ++i) {
        stringcase arg (argv[i] ? argv[i] : 0);
        if ((arg == "-i") || (arg == "/i") ||
            (arg == "-o") || (arg == "/o")) {
                ++i;
                continue;
        }
        fprintf (stderr, " %s", arg.c_str());
    }
    fprintf (stderr, "\n");

    // Parse input
    if (!tpyfile.parse (inpf)) {
        fprintf (stderr, "Unable to parse %s.\n", inpfilename.c_str());
        return 1;
    }

    // generate listing
    if (listing) {
        if (!tpyfile.process_symbols (listproc)) {
            fprintf (stderr, "Unable to generate listing.\n");
            return 1;
```

```
        }
        listproc.flush();
    }
    // generating epics db
    else {
        if (!tpyfile.process_symbols (dbproc)) {
            fprintf (stderr, "Unable to generate record database.\n");
            return 1;
        }
        dbproc.flush();
    }

    // write summary information
    split_io_support* summary= listing ?
        (split_io_support*)(&listproc) : (split_io_support*)(&dbproc);
    fprintf (outfilename.empty() ? stdout : stderr, "\nSummary:\n");
    fprintf (outfilename.empty() ? stdout : stderr,
        "Total number of processed records = %5d\n",
        summary->get_processed_total());
    fprintf (outfilename.empty() ? stdout : stderr,
        "Total number of input records     = %5d\n",
        summary->get_processed_readonly());
    fprintf (outfilename.empty() ? stdout : stderr,
        "Total number of in/out records    = %5d\n",
        summary->get_processed_io());

    return 0;
}
```

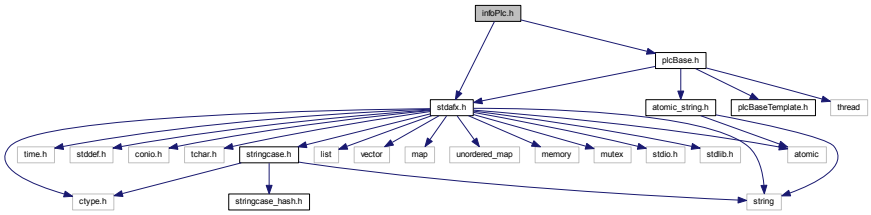
Here is the call graph for this function:



9.8 infoPlc.h File Reference

```
#include "stdafx.h"
#include "plcBase.h"
```

Include dependency graph for infoPlc.h:



Classes

- class InfoInterface

  Info interface.

- class InfoPLC

  Info plc.

9.8.1 Detailed Description

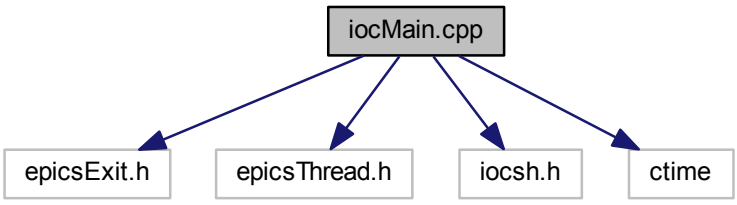
Header which includes classes for the info PLC.

Definition in file infoPlc.h.

9.9 iocMain.cpp File Reference

```
#include "epicsExit.h"
#include "epicsThread.h"
#include "iocsh.h"
#include <ctime>
```

Include dependency graph for iocMain.cpp:



Functions

- int **main** (int argc, char ∗argv[])

Variables

- clock_t **begin**
- clock_t **end**
- bool **debug**

9.9.1 Detailed Description

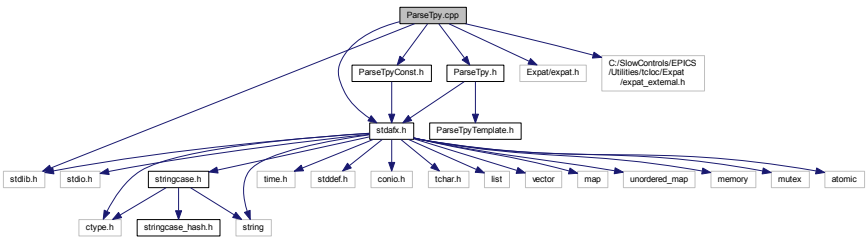
The main program for the TwinCAT IOC.

Definition in file iocMain.cpp.

9.10 ParseTpy.cpp File Reference

```
#include "stdafx.h"
#include "ParseTpyConst.h"
#include "ParseTpy.h"
#include "Expat/expat.h"
```

Include dependency graph for ParseTpy.cpp:



Classes

- class ParseTpy::parserinfo_type

Namespaces

- ParseTpy

  Namespace for tpy parsing.

Constant Groups

- ParseTpy

  Namespace for tpy parsing.

Macros

- #define **XML_STATIC**
- #define **XML_FMT_INT_MOD** "I"

Functions

- bool **ParseTpy::get_decoration** (const char **atts, unsigned int &decoration)

9.10.1 Detailed Description

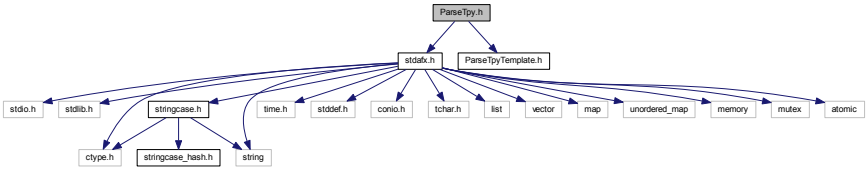
Methods to parse TwinCAT tpy file.

Definition in file ParseTpy.cpp.

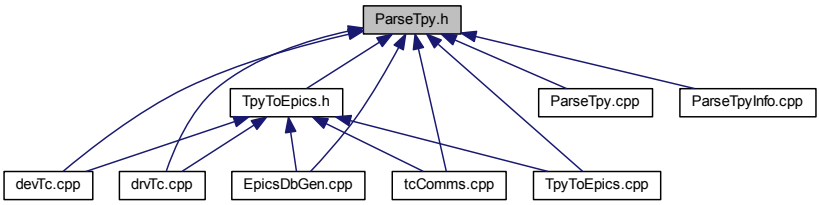
9.11 ParseTpy.h File Reference

```
#include "stdafx.h"
#include "ParseTpyTemplate.h"
```

Include dependency graph for ParseTpy.h:



This graph shows which files directly or indirectly include this file:



Classes

- class ParseTpy::opc_list
- class ParseTpy::ads_routing_info
- class ParseTpy::project_record
- class ParseTpy::base_record
- class ParseTpy::bit_location
- class ParseTpy::item_record
- class ParseTpy::type_record

- class ParseTpy::memory_location
- class ParseTpy::symbol_record
- class ParseTpy::process_arg
- class ParseTpy::tag_processing
- class ParseTpy::tpy_file

Namespaces

- ParseTpy

  Namespace for tpy parsing.

Constant Groups

- ParseTpy

  Namespace for tpy parsing.

Typedefs

- typedef std::map< int, std::stringcase > ParseTpy::property_map
- typedef std::pair< int, std::stringcase > ParseTpy::property_el
- typedef std::pair< int, int > ParseTpy::dimension
- typedef std::list< dimension > ParseTpy::dimensions
- typedef std::map< int, std::stringcase > ParseTpy::enum_map
- typedef std::pair< int, std::stringcase > ParseTpy::enum_pair
- typedef std::list< item_record > ParseTpy::item_list
- typedef std::map< unsigned int, type_record > ParseTpy::type_map
- typedef std::list< symbol_record > ParseTpy::symbol_list

Enumerations

- enum ParseTpy::opc_enum { ParseTpy::no_change, ParseTpy::publish, ParseTpy::silent }
- enum ParseTpy::type_enum { ParseTpy::unknown, ParseTpy::simple, ParseTpy::arraytype, ParseTpy::enumtype, ParseTpy::structtype, ParseTpy::functionblock }
- enum ParseTpy::process_type_enum { ParseTpy::pt_invalid, ParseTpy::pt_int, ParseTpy::pt_real, ParseTpy::pt_bool, ParseTpy::pt_string, ParseTpy::pt_enum, ParseTpy::pt_binary }
- enum ParseTpy::process_tag_enum { ParseTpy::process_all, ParseTpy::process_atomic, ParseTpy::process_structured }

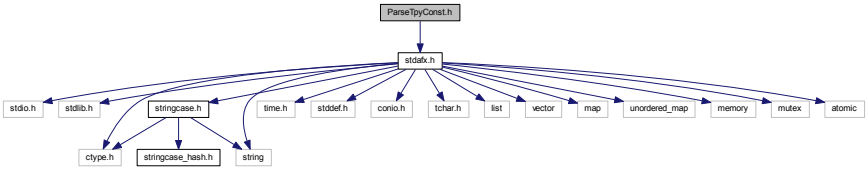
9.11.1 Detailed Description

Header which includes classes to parse a TwinCAT tpy file. It includes "ParseTpyConst.h"

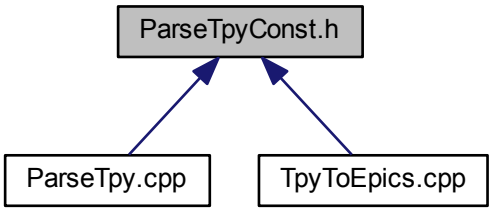
Definition in file ParseTpy.h.

9.12 ParseTpyConst.h File Reference

`#include "stdafx.h"`
Include dependency graph for ParseTpyConst.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- ParseTpy

  Namespace for tpy parsing.

Constant Groups

- ParseTpy

  Namespace for tpy parsing.

Variables

- const char ∗const **ParseTpy::xmlPlcProjectInfo** = "PlcProjectInfo"
- const char ∗const **ParseTpy::xmlProjectInfo** = "ProjectInfo"
- const char ∗const **ParseTpy::xmlRoutingInfo** = "RoutingInfo"
- const char ∗const **ParseTpy::xmlAdsInfo** = "AdsInfo"
- const char ∗const **ParseTpy::xmlDataTypes** = "DataTypes"
- const char ∗const **ParseTpy::xmlDataType** = "DataType"
- const char ∗const **ParseTpy::xmlSymbols** = "Symbols"
- const char ∗const **ParseTpy::xmlSymbol** = "Symbol"
- const char ∗const **ParseTpy::xmlProperties** = "Properties"
- const char ∗const **ParseTpy::xmlProperty** = "Property"

- const char *const **ParseTpy::xmlNetId** = "NetId"
- const char *const **ParseTpy::xmlPort** = "Port"
- const char *const **ParseTpy::xmlName** = "Name"
- const char *const **ParseTpy::xmlType** = "Type"
- const char *const **ParseTpy::xmlAttrDecoration** = "Decoration"
- const char *const **ParseTpy::xmlIGroup** = "IGroup"
- const char *const **ParseTpy::xmlIOffset** = "IOffset"
- const char *const **ParseTpy::xmlBitSize** = "BitSize"
- const char *const **ParseTpy::xmlBitOffs** = "BitOffs"
- const char *const **ParseTpy::xmlArrayInfo** = "ArrayInfo"
- const char *const **ParseTpy::xmlArrayLBound** = "LBound"
- const char *const **ParseTpy::xmlArrayElements** = "Elements"
- const char *const **ParseTpy::xmlSubItem** = "SubItem"
- const char *const **ParseTpy::xmlFbInfo** = "FbInfo"
- const char *const **ParseTpy::xmlEnumInfo** = "EnumInfo"
- const char *const **ParseTpy::xmlEnumText** = "Text"
- const char *const **ParseTpy::xmlEnumEnum** = "Enum"
- const char *const **ParseTpy::xmlValue** = "Value"
- const char *const **ParseTpy::xmlDesc** = "Desc"
- const char *const **ParseTpy::opcExport** = "opc"
- const char *const **ParseTpy::opcProp** = "opc_prop"
- const char *const **ParseTpy::opcBracket** = "["
- const int **ParseTpy::OPC_PROP_CDT** = 1
- const int **ParseTpy::OPC_PROP_VALUE** = 2
- const int **ParseTpy::OPC_PROP_QUALITY** = 3
- const int **ParseTpy::OPC_PROP_TIME** = 4
- const int **ParseTpy::OPC_PROP_RIGHTS** = 5
- const int **ParseTpy::OPC_PROP_SCANRATE** = 6
- const int **ParseTpy::OPC_PROP_UNIT** = 100
- const int **ParseTpy::OPC_PROP_DESC** = 101
- const int **ParseTpy::OPC_PROP_HIEU** = 102
- const int **ParseTpy::OPC_PROP_LOEU** = 103
- const int **ParseTpy::OPC_PROP_HIRANGE** = 104
- const int **ParseTpy::OPC_PROP_LORANGE** = 105
- const int **ParseTpy::OPC_PROP_CLOSE** = 106
- const int **ParseTpy::OPC_PROP_OPEN** = 107
- const int **ParseTpy::OPC_PROP_TIMEZONE** = 108
- const int **ParseTpy::OPC_PROP_FGC** = 201
- const int **ParseTpy::OPC_PROP_BGC** = 202
- const int **ParseTpy::OPC_PROP_BLINK** = 203
- const int **ParseTpy::OPC_PROP_BMP** = 204
- const int **ParseTpy::OPC_PROP_SND** = 205
- const int **ParseTpy::OPC_PROP_HTML** = 206
- const int **ParseTpy::OPC_PROP_AVI** = 207
- const int **ParseTpy::OPC_PROP_ALMSTAT** = 300
- const int **ParseTpy::OPC_PROP_ALMHELP** = 301
- const int **ParseTpy::OPC_PROP_ALMAREAS** = 302
- const int **ParseTpy::OPC_PROP_ALMPRIMARYAREA** = 303
- const int **ParseTpy::OPC_PROP_ALMCONDITION** = 304
- const int **ParseTpy::OPC_PROP_ALMLIMIT** = 305
- const int **ParseTpy::OPC_PROP_ALMDB** = 306
- const int **ParseTpy::OPC_PROP_ALMHH** = 307
- const int **ParseTpy::OPC_PROP_ALMH** = 308
- const int **ParseTpy::OPC_PROP_ALML** = 309
- const int **ParseTpy::OPC_PROP_ALMLL** = 310

- const int **ParseTpy::OPC_PROP_ALMROC** = 311
- const int **ParseTpy::OPC_PROP_ALMDEV** = 312
- const int **ParseTpy::OPC_PROP_PREC** = 8500
- const int **ParseTpy::OPC_PROP_ZRST** = 8510
- const int **ParseTpy::OPC_PROP_FFST** = 8525
- const int **ParseTpy::OPC_PROP_RECTYPE** = 8600
- const int **ParseTpy::OPC_PROP_INOUT** = 8601
- const char ∗const **ParseTpy::OPC_PROP_INPUT** = "input"
- const char ∗const **ParseTpy::OPC_PROP_OUTPUT** = "output"
- const int **ParseTpy::OPC_PROP_TSE** = 8602
- const int **ParseTpy::OPC_PROP_PINI** = 8603
- const int **ParseTpy::OPC_PROP_DTYP** = 8604
- const int **ParseTpy::OPC_PROP_SERVER** = 8610
- const int **ParseTpy::OPC_PROP_PLCNAME** = 8611
- const int **ParseTpy::OPC_PROP_ALMOSV** = 8700
- const int **ParseTpy::OPC_PROP_ALMZSV** = 8701
- const int **ParseTpy::OPC_PROP_ALMCOSV** = 8702
- const int **ParseTpy::OPC_PROP_ALMUNSV** = 8703
- const int **ParseTpy::OPC_PROP_ALMZRSV** = 8710
- const int **ParseTpy::OPC_PROP_ALMFFSV** = 8725

9.12.1 Detailed Description

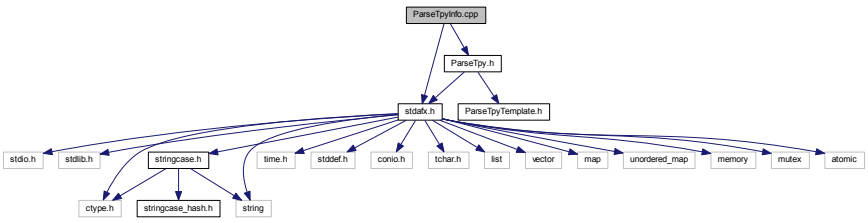
Header which includes constants needed to parse a TwinCAT tpy file.

Definition in file ParseTpyConst.h.

9.13 ParseTpyInfo.cpp File Reference

```
#include "stdafx.h"
#include "ParseTpy.h"
```

Include dependency graph for ParseTpyInfo.cpp:



Classes

- class syminfo_processing

Functions

- int main (int argc, char ∗argv[])

9.13.1 Detailed Description

Source file for tpy parsing methods.

Definition in file ParseTpyInfo.cpp.

9.13.2 Function Documentation

9.13.2.1 int main (int *argc*, char ∗ *argv[]*)

Main program

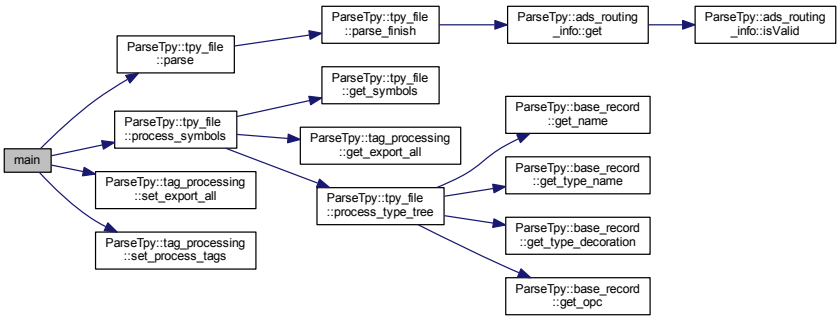
Definition at line 84 of file ParseTpyInfo.cpp.

References ParseTpy::tpy_file::parse(), ParseTpy::process_all, ParseTpy::process_atomic, ParseTpy::process_structured, ParseTpy::tpy_file::process_symbols(), ParseTpy::tag_processing::set_export_all(), and ParseTpy::tag_processing::set_process_tags().

```
85 {
86     int         help = 0;
87     bool        allsym = false;
88     bool        atomic = false;
89     bool        structured = false;
90     stringcase    prefix;
91     stringcase    inpfilename;
92     stringcase    outfilename;
93     tpy_file    tpyfile;
94
95     // command line parsing
96     for (int i = 1; i < argc; ++i) {
97         stringcase arg (argv[i] ? argv[i] : "");
98         stringcase next (i + 1 < argc && argv[i+1] ? argv[i+1] : "");
99         if (arg == "-h" || arg == "/h") {
100             help = 1;
101         }
102         else if (arg == "-l" || arg == "/l") {
103             allsym = true;
104         }
105         else if (arg == "-a" || arg == "/a") {
106             atomic = true;
107             structured = false;
108         }
109         else if (arg == "-s" || arg == "/s") {
110             atomic = false;
111             structured = true;
112         }
113         else if ((arg == "-p" || arg == "/p") && i + 1 < argc) {
114             prefix = next;
115             i += 1;
116         }
117         else if ((arg == "-i" || arg == "/i") && i + 1 < argc) {
118             inpfilename = next;
119             i += 1;
120         }
121         else if ((arg == "-o" || arg == "/o") && i + 1 < argc) {
122             outfilename = next;
123             i += 1;
124         }
125         else {
126             help = 2;
127         }
128     }
129     if (help) {
130         printf ("Usage: tpyinfo [-l|a] [p 'prefix'] -i 'tpyfile' -o 'outfile'\n"
131             "       Displays information about the individual symbols of a TwinCAT tpy file.\n"
132             "       -l includes all symbols rather than the ones available by opc\n"
133             "       -a only output atomic types\n"
134             "       -s only output structured types\n"
135             "       -p 'prefix' Add a prefix to the variable name\n"
136             "       -i 'tpyfile' input file name (stdin when omitted)\n"
137             "       -o 'outfile' output listing (stdout when omitted)\n");
138         if (help == 2) return 1; else return 0;
139     }
140
141     // open input file
142     FILE* inpf = stdin;
143     FILE* outf = stdout;
144     if (!inpfilename.empty()) {
145         inpf = fopen (inpfilename.c_str(), "r");
```

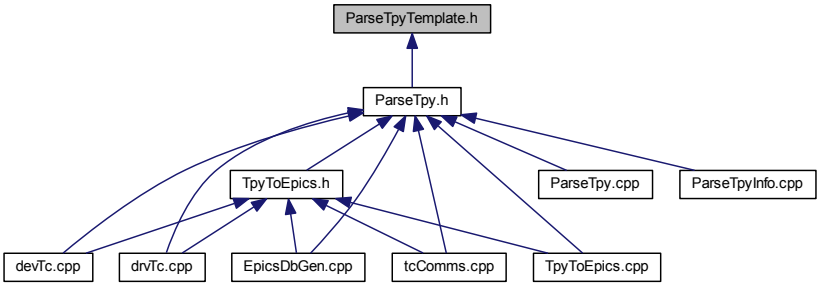
```
        if (!inpf) {
            fprintf (stderr, "Failed to open input %s.\n", inpfilename.c_str());
            return 1;
        }
    }
    // open output file
    if (!outfilename.empty()) {
        outf = fopen (outfilename.c_str(), "w");
        if (!outf) {
            fprintf (stderr, "Failed to open output %s.\n", outfilename.c_str());
            return 1;
        }
    }

    // parse tpy file
    if (!tpyfile.parse (inpf)) {
        fprintf (stderr, "Unable to parse %s\n", inpfilename.c_str());
        return 1;
    }

    // work through the symbol list
    tpyfile.set_export_all (allsym);
    tpyfile.set_process_tags (atomic ? process_atomic :
                              (structured ? process_structured :
    process_all));
    if (!tpyfile.process_symbols (syminfo_processing (outf), prefix)) {
        fprintf (stderr, "Unable to parse %s\n", inpfilename.c_str());
        return 1;
    }

    // close files
    if (!inpfilename.empty()) {
        fclose (inpf);
    }
    if (!outfilename.empty()) {
        fclose (outf);
    }
    return 0;
}
```

Here is the call graph for this function:



9.14 ParseTpyTemplate.h File Reference

This graph shows which files directly or indirectly include this file:



Namespaces

- ParseTpy

  Namespace for tpy parsing.

Constant Groups

- ParseTpy

  Namespace for tpy parsing.

9.14.1 Detailed Description

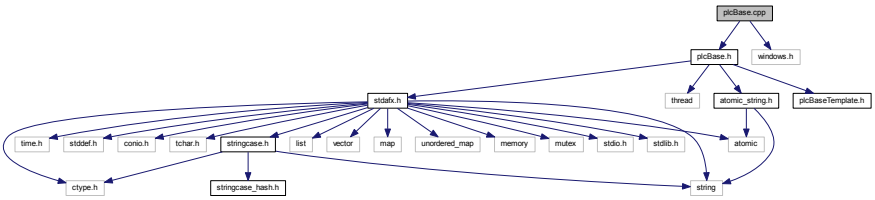
Templates for functions for parsing tpy file.

Definition in file ParseTpyTemplate.h.

9.15 plcBase.cpp File Reference

```
#include "plcBase.h"
#include <windows.h>
```

Include dependency graph for plcBase.cpp:



Classes

- struct plc::scanner_thread_args

Namespaces

- plc

  Namespace for abstract plc functionality.

Constant Groups

- plc

  Namespace for abstract plc functionality.

Functions

- template<>
  bool **plc::reset_and_read** (DataValueTypeDef::atomic_bool &dirty, DataValueTypeDef::type_wstring &dest, DataValueTypeDef::atomic_string *source)
- template<>
  bool **plc::write_and_test** (DataValueTypeDef::atomic_bool &dirty, DataValueTypeDef::atomic_bool &read_pending, DataValueTypeDef::atomic_wstring *dest, const DataValueTypeDef::type_string &source)
- VOID CALLBACK **plc::ScannerProc** (LPVOID lpArg, DWORD dwTimerLowValue, DWORD dwTimerHighValue)
- DWORD WINAPI **plc::scannerThread** (scanner_thread_args args)

9.15.1 Detailed Description

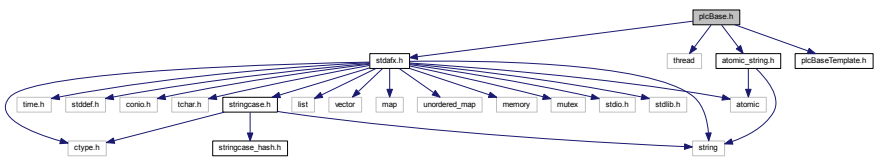
Defines methods for the internal record entry.

Definition in file plcBase.cpp.

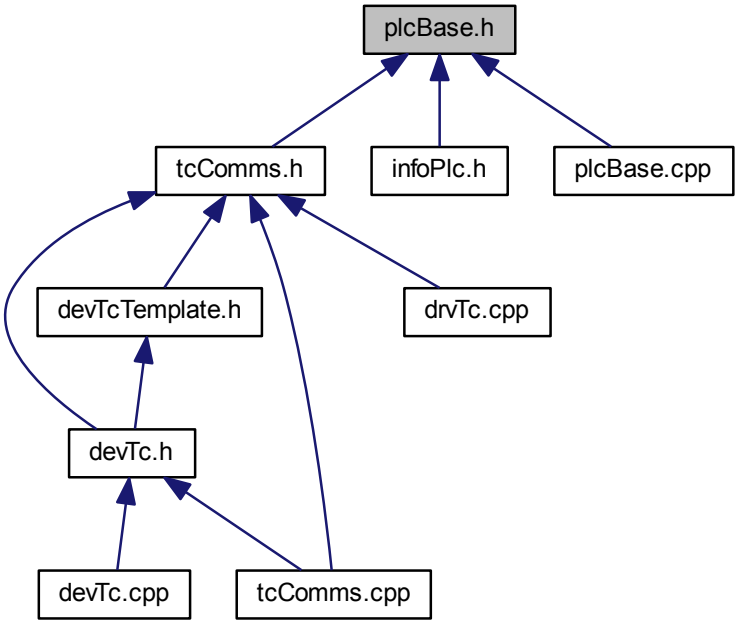
9.16 plcBase.h File Reference

```
#include "stdafx.h"
#include <thread>
#include "atomic_string.h"
#include "plcBaseTemplate.h"
```

Include dependency graph for plcBase.h:



This graph shows which files directly or indirectly include this file:



Classes

- class plc::Interface

  abstract interface
- struct plc::DataValueTraits< T >

  data value traits
- struct plc::DataValueTypeDef

  collection of type definitions
- class plc::DataValue

  data value
- class plc::BaseRecord

  Class for managing a tag/channel.
- class plc::BasePLC

  base PLC
- class plc::System

  System.

Namespaces

- plc

  Namespace for abstract plc functionality.

Constant Groups

- plc

  Namespace for abstract plc functionality.

Typedefs

- typedef std::unique_ptr < Interface > plc::InterfacePtr

  smart pointer to interface

- typedef std::shared_ptr < BaseRecord > plc::BaseRecordPtr

  smart pointer to record

- typedef std::unordered_map < std::stringcase, BaseRecordPtr > plc::BaseRecordList

  list of record

- typedef std::shared_ptr< BasePLC > plc::BasePLCPtr

  smart pointer to PLC

- typedef std::map < std::stringcase, BasePLCPtr > plc::BasePLCList

  BasePLC map.

Enumerations

- enum plc::data_type_enum {
  plc::dtInvalid, plc::dtBool, plc::dtInt8, plc::dtUInt8,
  plc::dtInt16, plc::dtUInt16, plc::dtInt32, plc::dtUInt32,
  plc::dtInt64, plc::dtUInt64, plc::dtFloat, plc::dtDouble,
  plc::dtString, plc::dtWString, plc::dtBinary }

  data type enumeration

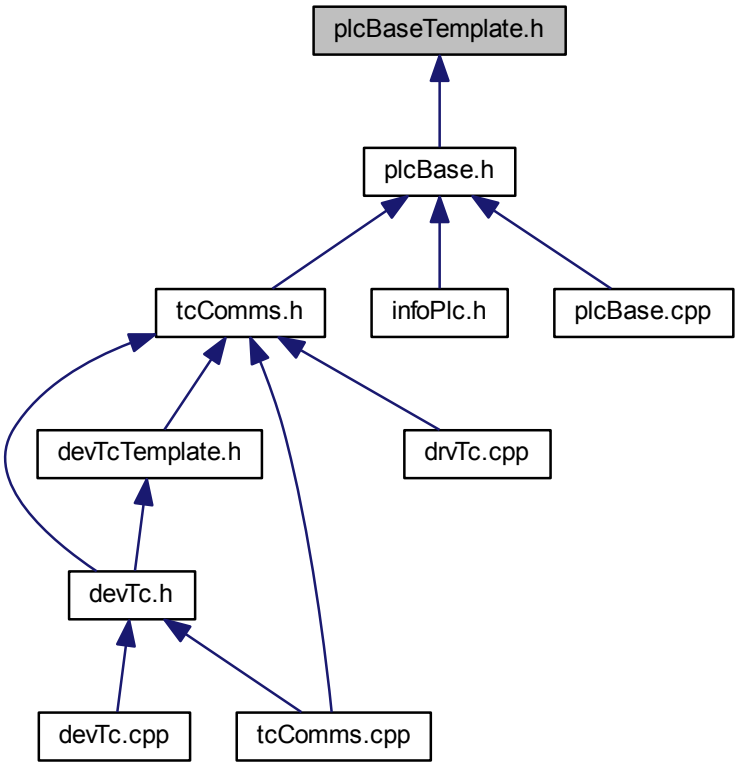
9.16.1 Detailed Description

Header which includes abstract base classes for defining an internal record entry for the IOC.

Definition in file plcBase.h.

9.17 plcBaseTemplate.h File Reference

This graph shows which files directly or indirectly include this file:



Namespaces

- plc

  Namespace for abstract plc functionality.

Constant Groups

- plc

  Namespace for abstract plc functionality.

Functions

- template<typename T , typename U >
  bool **plc::reset_and_read** (DataValueTypeDef::atomic_bool &dirty, T &dest, U *source)
- template<typename T >
  bool **plc::reset_and_read** (DataValueTypeDef::atomic_bool &dirty, T &dest, typename DataValueTraits< T >::traits_atomic *source)

- template<typename T , typename U >
  bool **plc::write_and_test** (DataValueTypeDef::atomic_bool &dirty, DataValueTypeDef::atomic_bool &read_-pending, U *dest, const T &source)
- template<typename T >
  bool **plc::write_and_test** (DataValueTypeDef::atomic_bool &dirty, DataValueTypeDef::atomic_bool &read_-pending, typename DataValueTraits< T >::traits_atomic *dest, const T &source)

9.17.1 Detailed Description

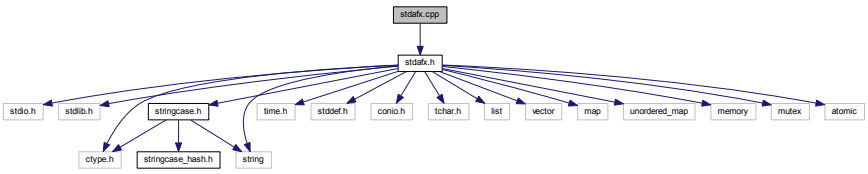
Header which includes templated methods for abstract record class and the DataValue class.

Definition in file plcBaseTemplate.h.

9.18 stdafx.cpp File Reference

```
#include "stdafx.h"
```

Include dependency graph for stdafx.cpp:



9.18.1 Detailed Description

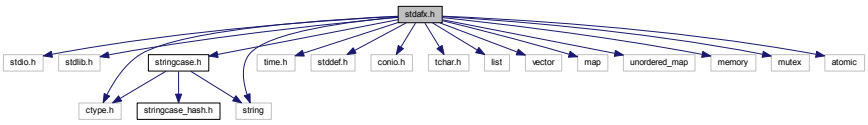
source file that includes just the standard includes EpicsDbGen.pch will be the pre-compiled header stdafx.obj will contain the pre-compiled type information

Definition in file stdafx.cpp.

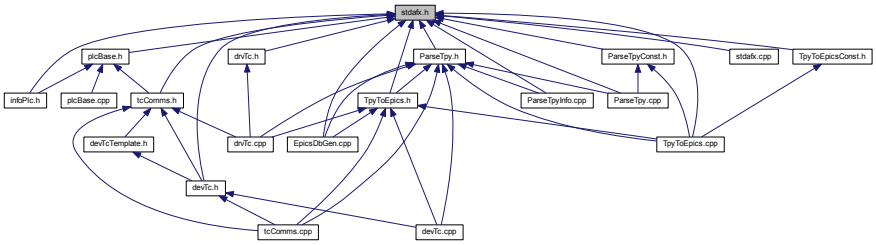
9.19 stdafx.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <time.h>
#include <stddef.h>
#include <conio.h>
#include <tchar.h>
#include <string>
#include <list>
#include <vector>
#include <map>
#include <unordered_map>
#include <memory>
#include <mutex>
#include <atomic>
#include "stringcase.h"
```

Include dependency graph for stdafx.h:



This graph shows which files directly or indirectly include this file:



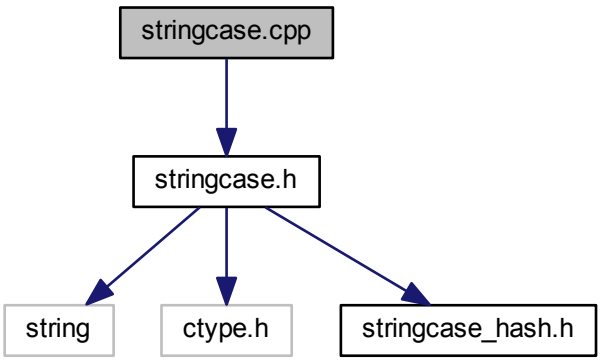
9.19.1 Detailed Description

Include file for standard system include files, or project specific include files that are used frequently, but are changed infrequently

Definition in file stdafx.h.

9.20 stringcase.cpp File Reference

`#include "stringcase.h"`
Include dependency graph for stringcase.cpp:



Functions

- int **std::_strncasecmp** (const char ∗s1, const char ∗s2, int n)
- int **std::_wcsncasewcmp** (const wchar_t ∗s1, const wchar_t ∗s2, int n)
- void std::trim_space (stringcase &s)
- void std::trim_space (wstringcase &s)

9.20.1 Detailed Description

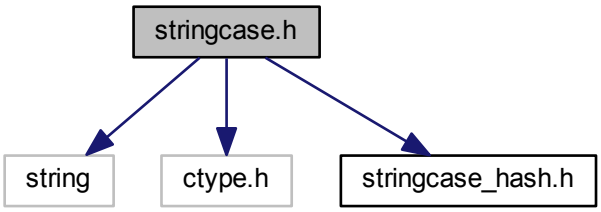
Methods for case insensitve strings.

Definition in file stringcase.cpp.

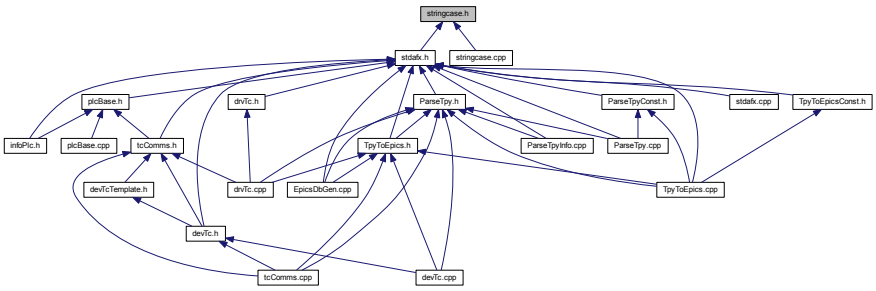
9.21 stringcase.h File Reference

```
#include <string>
#include <ctype.h>
#include "stringcase_hash.h"
```

Include dependency graph for stringcase.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct std::case_char_traits

  case insensitive traits.
- struct std::case_wchar_traits

case insensitive unicode traits.

Typedefs

- typedef std::basic_string
  < char, case_char_traits > std::stringcase

  case insensitive string.
- typedef std::basic_string
  < wchar_t, case_wchar_traits > std::wstringcase

  case insensitive string.

Functions

- int std::strncasecmp (const char ∗s1, const char ∗s2, int n)

  case insensitive compare with maximum length
- int std::wcsncasewcmp (const wchar_t ∗s1, const wchar_t ∗s2, int n)

  case insensitive unicode compare with maximum length
- void std::trim_space (stringcase &s)
- void std::trim_space (wstringcase &s)

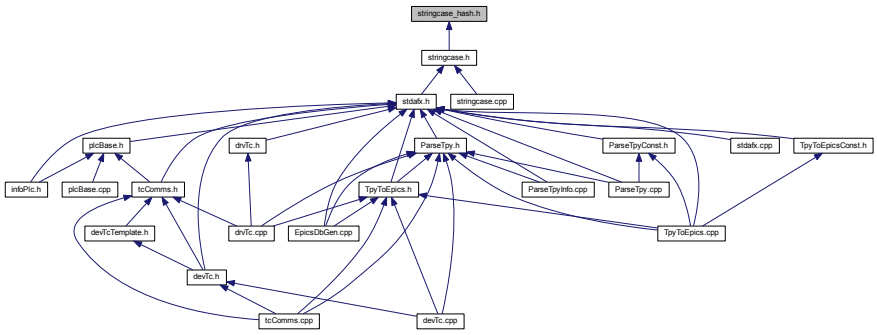
9.21.1 Detailed Description

Functions and classes for a case-insensitive string.

Definition in file stringcase.h.

9.22 stringcase_hash.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- struct std::std::hash< stringcase >

  hash for case insensitive string.
- struct std::std::hash< wstringcase >

  hash for case insensitive unicode string.

9.22.1 Detailed Description

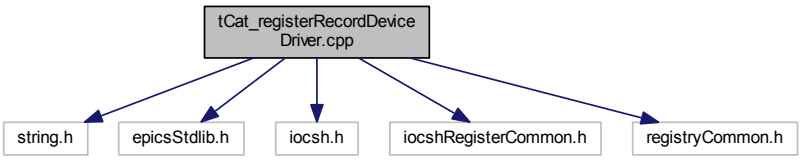
Specialization for case sensitive strings.

Definition in file stringcase_hash.h.

9.23 tCat_registerRecordDeviceDriver.cpp File Reference

```
#include <string.h>
#include "epicsStdlib.h"
#include "iocsh.h"
#include "iocshRegisterCommon.h"
#include "registryCommon.h"
```

Include dependency graph for tCat_registerRecordDeviceDriver.cpp:



Functions

- int **tCat_registerRecordDeviceDriver** (DBBASE ∗pbase)

Variables

- epicsShareExtern rset ∗ **pvar_rset_aaiRSET**
- epicsShareExtern int(∗ **pvar_func_aaiRecordSizeOffset**)(dbRecordType ∗pdbRecordType)
- epicsShareExtern rset ∗ **pvar_rset_aaoRSET**
- epicsShareExtern int(∗ **pvar_func_aaoRecordSizeOffset**)(dbRecordType ∗pdbRecordType)
- epicsShareExtern rset ∗ **pvar_rset_aiRSET**
- epicsShareExtern int(∗ **pvar_func_aiRecordSizeOffset**)(dbRecordType ∗pdbRecordType)
- epicsShareExtern rset ∗ **pvar_rset_aoRSET**
- epicsShareExtern int(∗ **pvar_func_aoRecordSizeOffset**)(dbRecordType ∗pdbRecordType)
- epicsShareExtern rset ∗ **pvar_rset_aSubRSET**
- epicsShareExtern int(∗ **pvar_func_aSubRecordSizeOffset**)(dbRecordType ∗pdbRecordType)
- epicsShareExtern rset ∗ **pvar_rset_biRSET**
- epicsShareExtern int(∗ **pvar_func_biRecordSizeOffset**)(dbRecordType ∗pdbRecordType)
- epicsShareExtern rset ∗ **pvar_rset_boRSET**
- epicsShareExtern int(∗ **pvar_func_boRecordSizeOffset**)(dbRecordType ∗pdbRecordType)
- epicsShareExtern rset ∗ **pvar_rset_calcRSET**
- epicsShareExtern int(∗ **pvar_func_calcRecordSizeOffset**)(dbRecordType ∗pdbRecordType)
- epicsShareExtern rset ∗ **pvar_rset_calcoutRSET**
- epicsShareExtern int(∗ **pvar_func_calcoutRecordSizeOffset**)(dbRecordType ∗pdbRecordType)
- epicsShareExtern rset ∗ **pvar_rset_compressRSET**
- epicsShareExtern int(∗ **pvar_func_compressRecordSizeOffset**)(dbRecordType ∗pdbRecordType)
- epicsShareExtern rset ∗ **pvar_rset_dfanoutRSET**
- epicsShareExtern int(∗ **pvar_func_dfanoutRecordSizeOffset**)(dbRecordType ∗pdbRecordType)

- epicsShareExtern rset ∗ **pvar_rset_eventRSET**
- epicsShareExtern int(∗ **pvar_func_eventRecordSizeOffset**)(dbRecordType ∗pdbRecordType)
- epicsShareExtern rset ∗ **pvar_rset_fanoutRSET**
- epicsShareExtern int(∗ **pvar_func_fanoutRecordSizeOffset**)(dbRecordType ∗pdbRecordType)
- epicsShareExtern rset ∗ **pvar_rset_longinRSET**
- epicsShareExtern int(∗ **pvar_func_longinRecordSizeOffset**)(dbRecordType ∗pdbRecordType)
- epicsShareExtern rset ∗ **pvar_rset_longoutRSET**
- epicsShareExtern int(∗ **pvar_func_longoutRecordSizeOffset**)(dbRecordType ∗pdbRecordType)
- epicsShareExtern rset ∗ **pvar_rset_mbbiRSET**
- epicsShareExtern int(∗ **pvar_func_mbbiRecordSizeOffset**)(dbRecordType ∗pdbRecordType)
- epicsShareExtern rset ∗ **pvar_rset_mbbiDirectRSET**
- epicsShareExtern int(∗ **pvar_func_mbbiDirectRecordSizeOffset**)(dbRecordType ∗pdbRecordType)
- epicsShareExtern rset ∗ **pvar_rset_mbboRSET**
- epicsShareExtern int(∗ **pvar_func_mbboRecordSizeOffset**)(dbRecordType ∗pdbRecordType)
- epicsShareExtern rset ∗ **pvar_rset_mbboDirectRSET**
- epicsShareExtern int(∗ **pvar_func_mbboDirectRecordSizeOffset**)(dbRecordType ∗pdbRecordType)
- epicsShareExtern rset ∗ **pvar_rset_permissiveRSET**
- epicsShareExtern int(∗ **pvar_func_permissiveRecordSizeOffset**)(dbRecordType ∗pdbRecordType)
- epicsShareExtern rset ∗ **pvar_rset_selRSET**
- epicsShareExtern int(∗ **pvar_func_selRecordSizeOffset**)(dbRecordType ∗pdbRecordType)
- epicsShareExtern rset ∗ **pvar_rset_seqRSET**
- epicsShareExtern int(∗ **pvar_func_seqRecordSizeOffset**)(dbRecordType ∗pdbRecordType)
- epicsShareExtern rset ∗ **pvar_rset_stateRSET**
- epicsShareExtern int(∗ **pvar_func_stateRecordSizeOffset**)(dbRecordType ∗pdbRecordType)
- epicsShareExtern rset ∗ **pvar_rset_stringinRSET**
- epicsShareExtern int(∗ **pvar_func_stringinRecordSizeOffset**)(dbRecordType ∗pdbRecordType)
- epicsShareExtern rset ∗ **pvar_rset_stringoutRSET**
- epicsShareExtern int(∗ **pvar_func_stringoutRecordSizeOffset**)(dbRecordType ∗pdbRecordType)
- epicsShareExtern rset ∗ **pvar_rset_subRSET**
- epicsShareExtern int(∗ **pvar_func_subRecordSizeOffset**)(dbRecordType ∗pdbRecordType)
- epicsShareExtern rset ∗ **pvar_rset_subArrayRSET**
- epicsShareExtern int(∗ **pvar_func_subArrayRecordSizeOffset**)(dbRecordType ∗pdbRecordType)
- epicsShareExtern rset ∗ **pvar_rset_waveformRSET**
- epicsShareExtern int(∗ **pvar_func_waveformRecordSizeOffset**)(dbRecordType ∗pdbRecordType)
- epicsShareExtern dset ∗ **pvar_dset_devAaiSoft**
- epicsShareExtern dset ∗ **pvar_dset_aaival_record_tc_dset**
- epicsShareExtern dset ∗ **pvar_dset_devAaoSoft**
- epicsShareExtern dset ∗ **pvar_dset_aaoval_record_tc_dset**
- epicsShareExtern dset ∗ **pvar_dset_devAiSoft**
- epicsShareExtern dset ∗ **pvar_dset_devAiSoftRaw**
- epicsShareExtern dset ∗ **pvar_dset_devTimestampAI**
- epicsShareExtern dset ∗ **pvar_dset_devAiGeneralTime**
- epicsShareExtern dset ∗ **pvar_dset_aival_record_tc_dset**
- epicsShareExtern dset ∗ **pvar_dset_airval_record_tc_dset**
- epicsShareExtern dset ∗ **pvar_dset_devAoSoft**
- epicsShareExtern dset ∗ **pvar_dset_devAoSoftRaw**
- epicsShareExtern dset ∗ **pvar_dset_devAoSoftCallback**
- epicsShareExtern dset ∗ **pvar_dset_aoval_record_tc_dset**
- epicsShareExtern dset ∗ **pvar_dset_aorval_record_tc_dset**
- epicsShareExtern dset ∗ **pvar_dset_devBiSoft**
- epicsShareExtern dset ∗ **pvar_dset_devBiSoftRaw**
- epicsShareExtern dset ∗ **pvar_dset_bival_record_tc_dset**
- epicsShareExtern dset ∗ **pvar_dset_birval_record_tc_dset**
- epicsShareExtern dset ∗ **pvar_dset_devBoSoft**
- epicsShareExtern dset ∗ **pvar_dset_devBoSoftRaw**

- epicsShareExtern dset * **pvar_dset_devBoSoftCallback**
- epicsShareExtern dset * **pvar_dset_devBoGeneralTime**
- epicsShareExtern dset * **pvar_dset_boval_record_tc_dset**
- epicsShareExtern dset * **pvar_dset_borval_record_tc_dset**
- epicsShareExtern dset * **pvar_dset_devCalcoutSoft**
- epicsShareExtern dset * **pvar_dset_devCalcoutSoftCallback**
- epicsShareExtern dset * **pvar_dset_devEventSoft**
- epicsShareExtern dset * **pvar_dset_devLiSoft**
- epicsShareExtern dset * **pvar_dset_devLiGeneralTime**
- epicsShareExtern dset * **pvar_dset_longinval_record_tc_dset**
- epicsShareExtern dset * **pvar_dset_devLoSoft**
- epicsShareExtern dset * **pvar_dset_devLoSoftCallback**
- epicsShareExtern dset * **pvar_dset_longoutval_record_tc_dset**
- epicsShareExtern dset * **pvar_dset_devMbbiSoft**
- epicsShareExtern dset * **pvar_dset_devMbbiSoftRaw**
- epicsShareExtern dset * **pvar_dset_mbbival_record_tc_dset**
- epicsShareExtern dset * **pvar_dset_mbbirval_record_tc_dset**
- epicsShareExtern dset * **pvar_dset_devMbbiDirectSoft**
- epicsShareExtern dset * **pvar_dset_devMbbiDirectSoftRaw**
- epicsShareExtern dset * **pvar_dset_mbbiDirectval_record_tc_dset**
- epicsShareExtern dset * **pvar_dset_mbbiDirectrval_record_tc_dset**
- epicsShareExtern dset * **pvar_dset_devMbboSoft**
- epicsShareExtern dset * **pvar_dset_devMbboSoftRaw**
- epicsShareExtern dset * **pvar_dset_devMbboSoftCallback**
- epicsShareExtern dset * **pvar_dset_mbboval_record_tc_dset**
- epicsShareExtern dset * **pvar_dset_mbborval_record_tc_dset**
- epicsShareExtern dset * **pvar_dset_devMbboDirectSoft**
- epicsShareExtern dset * **pvar_dset_devMbboDirectSoftRaw**
- epicsShareExtern dset * **pvar_dset_devMbboDirectSoftCallback**
- epicsShareExtern dset * **pvar_dset_mbboDirectval_record_tc_dset**
- epicsShareExtern dset * **pvar_dset_mbboDirectrval_record_tc_dset**
- epicsShareExtern dset * **pvar_dset_devSiSoft**
- epicsShareExtern dset * **pvar_dset_devTimestampSI**
- epicsShareExtern dset * **pvar_dset_devSiGeneralTime**
- epicsShareExtern dset * **pvar_dset_stringinval_record_tc_dset**
- epicsShareExtern dset * **pvar_dset_devSoSoft**
- epicsShareExtern dset * **pvar_dset_devSoSoftCallback**
- epicsShareExtern dset * **pvar_dset_devSoStdio**
- epicsShareExtern dset * **pvar_dset_stringoutval_record_tc_dset**
- epicsShareExtern dset * **pvar_dset_devSASoft**
- epicsShareExtern dset * **pvar_dset_devWfSoft**
- epicsShareExtern dset * **pvar_dset_waveformval_record_tc_dset**
- epicsShareExtern void(* **pvar_func_asSub**)(void)
- epicsShareExtern int * **pvar_int_asCaDebug**
- epicsShareExtern int * **pvar_int_dbRecordsOnceOnly**
- epicsShareExtern int * **pvar_int_dbBptNotMonotonic**

9.23.1 Detailed Description

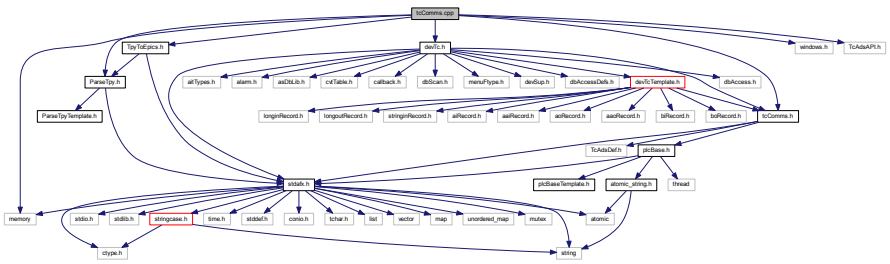
Generated from ../O.Common/tCat.dbd

Definition in file tCat_registerRecordDeviceDriver.cpp.

9.24 tcComms.cpp File Reference

```
#include "tcComms.h"
#include "devTc.h"
#include "ParseTpy.h"
#include "windows.h"
#include "TpyToEpics.h"
#include "TcAdsAPI.h"
#include <memory>
```

Include dependency graph for tcComms.cpp:



Namespaces

- TcComms

  Namespace for TCat communication.

Constant Groups

- TcComms

  Namespace for TCat communication.

Typedefs

- typedef std::auto_ptr< BasePLC > **TcComms::plcPtr**

Functions

- void TcComms::errorPrintf (int nErr)
- bool **TcComms::compByOffset** (BaseRecordPtr recA, BaseRecordPtr recB)

Variables

- bool **debug** = true
- bool **tcdebug** = false

9.24.1 Detailed Description

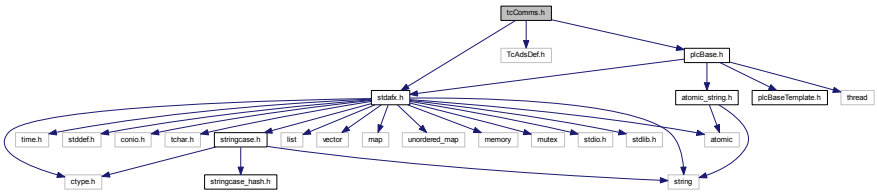
Defines methods for TwinCAT communication.

Definition in file tcComms.cpp.

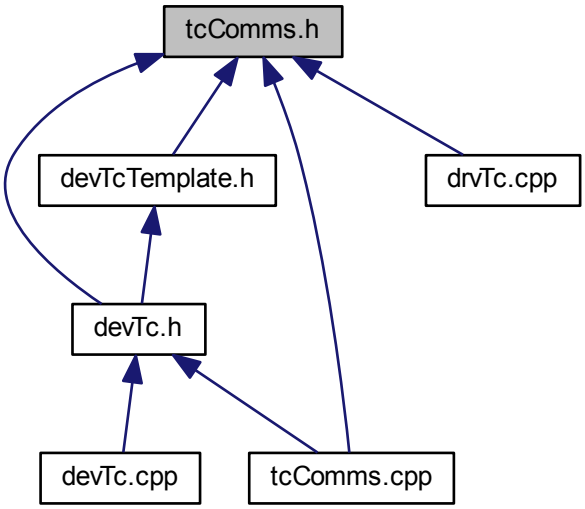
9.25 tcComms.h File Reference

```
#include "stdafx.h"
#include "TcAdsDef.h"
#include "plcBase.h"
```

Include dependency graph for tcComms.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct TcComms::DataPar

  Memory location struct.
- class TcComms::TCatInterface

  TCat interface class.
- class TcComms::tcProcWrite

  TwinCAT process write requests.
- class TcComms::TcPLC

  TwinCAT PLC.

Namespaces

- TcComms

  Namespace for TCat communication.

Constant Groups

- TcComms

  Namespace for TCat communication.

Macros

- #define MAX_REQ_SIZE 250000

  maximum allowed request size (bytes)
- #define MAX_SINGLE_GAP_SIZE 50

  maximum allowed size (bytes) of a memory gap within continuous request
- #define MAX_REL_GAP 0.25

  (maximum allowed total gap size) / (current request size)
- #define MIN_REL_GAP_SIZE 100

  minimum allowed relative gap size (bytes)

Variables

- const int default_scanrate = 100

  default PLC TwinCAT scan rate (100ms)
- const int minimum_scanrate = 5

  minimum PLC TwinCAT scan rate (5ms)
- const int maximum_scanrate = 10000

  maximum PLC TwinCAT scan rate (10s)
- const int default_multiple = 10

  default multiple for PLC EPICS scan rate (10)
- const int minimum_multiple = 1

  minimum multiple for PLC EPICS scan rate (1)
- const int maximum_multiple = 200

  maximum multiple for PLC EPICS scan rate (200)

9.25.1 Detailed Description

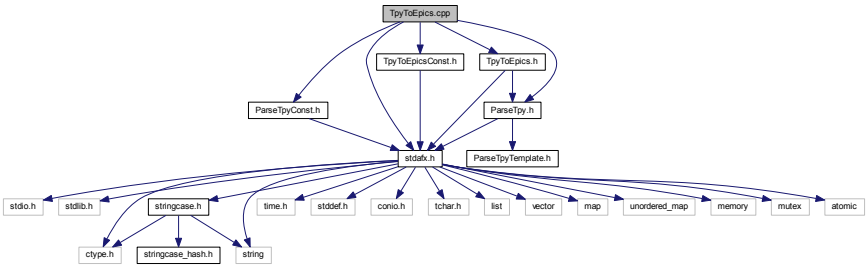
Header which includes classes to interface with the TCat system and manage TCat symbols.

Definition in file tcComms.h.

9.26 TpyToEpics.cpp File Reference

```
#include "stdafx.h"
#include "ParseTpyConst.h"
#include "ParseTpy.h"
#include "TpyToEpicsConst.h"
#include "TpyToEpics.h"
```

Include dependency graph for TpyToEpics.cpp:



Namespaces

- EpicsTpy

  Namespace for tpy-db conversion.

Constant Groups

- EpicsTpy

  Namespace for tpy-db conversion.

9.26.1 Detailed Description

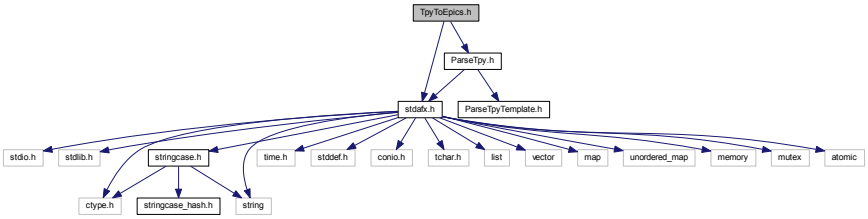
Source for methods that generate EPICs .db file from a .tpy file

Definition in file TpyToEpics.cpp.

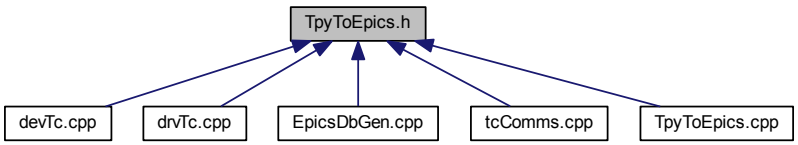
9.27 TpyToEpics.h File Reference

```
#include "stdafx.h"
#include "ParseTpy.h"
```

Include dependency graph for TpyToEpics.h:



This graph shows which files directly or indirectly include this file:



Classes

- class EpicsTpy::epics_conversion
- class EpicsTpy::split_io_support
- class EpicsTpy::epics_list_processing
- class EpicsTpy::epics_db_processing

Namespaces

- EpicsTpy

  Namespace for tpy-db conversion.

Constant Groups

- EpicsTpy

  Namespace for tpy-db conversion.

Enumerations

- enum EpicsTpy::tc_epics_conv { EpicsTpy::no_conversion, EpicsTpy::no_dot, EpicsTpy::ligo_std }
- enum EpicsTpy::case_type { EpicsTpy::preserve_case, EpicsTpy::upper_case, EpicsTpy::lower_case }
- enum EpicsTpy::listing_type { EpicsTpy::listing_standard, EpicsTpy::listing_epics, EpicsTpy::listing_autoburt }
- enum EpicsTpy::device_support_type { EpicsTpy::device_support_opc_name, EpicsTpy::device_support_tc_name }

9.27.1 Detailed Description

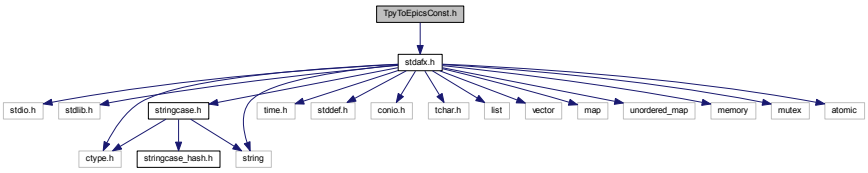
Header which includes classes to convert a parsed TwinCAT tpy into an EPICS database. It includes "ParseTpy.h"

Definition in file TpyToEpics.h.

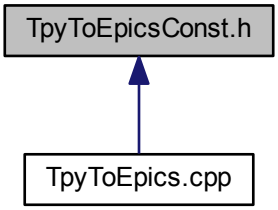
9.28 TpyToEpicsConst.h File Reference

```
#include "stdafx.h"
```

Include dependency graph for TpyToEpicsConst.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- ParseTpy

  Namespace for tpy parsing.
- EpicsTpy

  Namespace for tpy-db conversion.

Constant Groups

- ParseTpy

  Namespace for tpy parsing.
- EpicsTpy

  Namespace for tpy-db conversion.

Variables

- const int **EpicsTpy::MAX_EPICS_CHANNEL** = 54
- const int **EpicsTpy::MAX_EPICS_DESC** = 40
- const int **EpicsTpy::MAX_EPICS_UNIT** = 15
- const char ∗const **EpicsTpy::EPICS_DB_EGU** = "EGU"
- const char ∗const **EpicsTpy::EPICS_DB_DESC** = "DESC"
- const char ∗const **EpicsTpy::EPICS_DB_HOPR** = "HOPR"
- const char ∗const **EpicsTpy::EPICS_DB_LOPR** = "LOPR"
- const char ∗const **EpicsTpy::EPICS_DB_DRVH** = "DRVH"
- const char ∗const **EpicsTpy::EPICS_DB_DRVL** = "DRVL"

- const char *const **EpicsTpy::EPICS_DB_ONAM** = "ONAM"
- const char *const **EpicsTpy::EPICS_DB_ZNAM** = "ZNAM"
- const char *const **EpicsTpy::EPICS_DB_PREC** = "PREC"
- const char *const **EpicsTpy::EPICS_DB_ZRST** [16]
- const char *const **EpicsTpy::EPICS_DB_ZRVL** [16]
- const char *const **EpicsTpy::EPICS_DB_SCAN** = "SCAN"
- const char *const **EpicsTpy::EPICS_DB_INP** = "INP"
- const char *const **EpicsTpy::EPICS_DB_OUT** = "OUT"
- const char *const **EpicsTpy::EPICS_DB_TSE** = "TSE"
- const char *const **EpicsTpy::EPICS_DB_PINI** = "PINI"
- const char *const **EpicsTpy::EPICS_DB_DTYP** = "DTYP"
- const char *const **EpicsTpy::EPICS_DB_OSV** = "OSV"
- const char *const **EpicsTpy::EPICS_DB_ZSV** = "ZSV"
- const char *const **EpicsTpy::EPICS_DB_COSV** = "COSV"
- const char *const **EpicsTpy::EPICS_DB_HIHI** = "HIHI"
- const char *const **EpicsTpy::EPICS_DB_HIGH** = "HIGH"
- const char *const **EpicsTpy::EPICS_DB_LOW** = "LOW"
- const char *const **EpicsTpy::EPICS_DB_LOLO** = "LOLO"
- const char *const **EpicsTpy::EPICS_DB_HYST** = "HYST"
- const char *const **EpicsTpy::EPICS_DB_HHSV** = "HHSV"
- const char *const **EpicsTpy::EPICS_DB_HSV** = "HSV"
- const char *const **EpicsTpy::EPICS_DB_LSV** = "LSV"
- const char *const **EpicsTpy::EPICS_DB_LLSV** = "LLSV"
- const char *const **EpicsTpy::EPICS_DB_NOALARM** = "NO_ALARM"
- const char *const **EpicsTpy::EPICS_DB_MINOR** = "MINOR"
- const char *const **EpicsTpy::EPICS_DB_MAJOR** = "MAJOR"
- const char *const **EpicsTpy::EPICS_DB_UNSV** = "UNSV"
- const char *const **EpicsTpy::EPICS_DB_ZRSV** [16]
- const char *const EpicsTpy::EPICS_DB_FORBIDDEN []
- const char *const EpicsTpy::EPICS_DB_ALLOWED []
- const char *const EpicsTpy::EPICS_DB_NUMVAL []

9.28.1 Detailed Description

Header which includes EPICS constants needed to generate an EPICS db file fromt a parsed TwinCAT tpy file.

Definition in file TpyToEpicsConst.h.

Index

~InfoPLC
    InfoPLC, 166

aaival
    DevTc, 39
aaoval
    DevTc, 39
add
    plc::BasePLC, 63
    plc::System, 190
    TcComms::tcProcWrite, 207
airval
    DevTc, 39
aival
    DevTc, 39
aorval
    DevTc, 39
aoval
    DevTc, 39
arraytype
    Classes for describing a type, 16
atomic_string.h, 223

base_record
    ParseTpy::base_record, 59, 60
BasePLCList
    plc, 45
BasePLCPtr
    plc, 45
BaseRecord
    plc::BaseRecord, 69
BaseRecordList
    plc, 46
BaseRecordPtr
    plc, 46
birval
    DevTc, 39
bival
    DevTc, 39
borval
    DevTc, 39
boval
    DevTc, 39

callbackRequestPending
    DevTc::EpicsInterface, 161
case_type
    Utility functions and classes, 30
Classes for converting a paresed tpy, 31
    device_support_opc_name, 31
    device_support_tc_name, 31
    device_support_type, 31
    listing_autoburt, 31
    listing_epics, 31
    listing_standard, 31
    listing_type, 31
Classes for describing a structure element, 15
    dimension, 15
    dimensions, 15
    enum_map, 15
    enum_pair, 15
    item_list, 15
Classes for describing a symbol, 17
    symbol_list, 17
Classes for describing a type, 16
    arraytype, 16
    enumtype, 16
    functionblock, 16
    simple, 16
    structtype, 16
    type_enum, 16
    type_map, 16
    unknown, 16
Classes for describing TC symbol, 28
Classes for describing the parser, 18
    process_all, 18
    process_atomic, 18
    process_structured, 18
    process_tag_enum, 18
    process_type_enum, 18
    pt_binary, 18
    pt_bool, 18
    pt_enum, 18
    pt_int, 18
    pt_invalid, 18
    pt_real, 18
    pt_string, 18
Classes for managing groups of TC symbols, 29
closePort
    TcComms::TcPLC, 203
Constants related to read/write scanning, 27

data_type_enum
    plc, 46
DataValue
    plc::DataValue, 86
DevTc
    aaival, 39
    aaoval, 39
    airval, 39

aival, 39
aorval, 39
aoval, 39
birval, 39
bival, 39
borval, 39
boval, 39
epics_record_enumEnd, 39
eventval, 39
histogramval, 39
invalidval, 39
longinval, 39
longoutval, 39
mbbiDirectrval, 39
mbbiDirectval, 39
mbbirval, 39
mbbival, 39
mbboDirectrval, 39
mbboDirectval, 39
mbborval, 39
mbboval, 39
stringinval, 39
stringoutval, 39
waveformval, 39
DevTc, 37
epics_record_enum, 39
devTc.cpp, 224
linkRecord, 226
devTc.h, 228
DevTc::EpicsInterface, 159
callbackRequestPending, 161
isCallback, 161
isPassive, 161
pull, 161
push, 161
DevTc::devTcDefIn< RecType >, 115
DevTc::devTcDefIo< RecType >, 116
DevTc::devTcDefOut< RecType >, 118
DevTc::devTcDefWaveformIn< RecType >, 119
DevTc::epics_record_traits
value_ait_type, 138
value_count, 138
DevTc::epics_record_traits< aaival >, 139
DevTc::epics_record_traits< aaoval >, 139
DevTc::epics_record_traits< airval >, 140
DevTc::epics_record_traits< aival >, 141
DevTc::epics_record_traits< aorval >, 141
DevTc::epics_record_traits< aoval >, 142
DevTc::epics_record_traits< birval >, 143
DevTc::epics_record_traits< bival >, 143
DevTc::epics_record_traits< borval >, 144
DevTc::epics_record_traits< boval >, 145
DevTc::epics_record_traits< eventval >, 145
DevTc::epics_record_traits< histogramval >, 146
DevTc::epics_record_traits< longinval >, 147
DevTc::epics_record_traits< longoutval >, 147
DevTc::epics_record_traits< mbbiDirectrval >, 148
DevTc::epics_record_traits< mbbiDirectval >, 149
DevTc::epics_record_traits< mbbirval >, 149
DevTc::epics_record_traits< mbbival >, 150
DevTc::epics_record_traits< mbboDirectrval >, 151
DevTc::epics_record_traits< mbboDirectval >, 151
DevTc::epics_record_traits< mbborval >, 152
DevTc::epics_record_traits< mbboval >, 153
DevTc::epics_record_traits< RecType >, 137
DevTc::epics_record_traits< RecType >::traits_type, 220
DevTc::epics_record_traits< stringinval >, 153
DevTc::epics_record_traits< stringoutval >, 154
DevTc::epics_record_traits< waveformval >, 155
DevTc::epics_tc_db_processing, 156
operator(), 157
DevTc::tcRegisterToIocShell, 208
devTcTemplate.h, 229
device_support_opc_name
Classes for converting a paresed tpy, 31
device_support_tc_name
Classes for converting a paresed tpy, 31
device_support_type
Classes for converting a paresed tpy, 31
dimension
Classes for describing a structure element, 15
dimensions
Classes for describing a structure element, 15
drvTc.cpp, 232
drvTc.h, 233
dtBinary
plc, 46
dtBool
plc, 46
dtDouble
plc, 46
dtFloat
plc, 46
dtInt16
plc, 46
dtInt32
plc, 46
dtInt64
plc, 46
dtInt8
plc, 46
dtInvalid
plc, 46
dtString
plc, 46
dtUInt16
plc, 46
dtUInt32
plc, 46
dtUInt64
plc, 46
dtUInt8
plc, 46
dtWString
plc, 46

EPICS maximum length constants, 32
EPICS record field names, 33
    EPICS_DB_ZRST, 33
    EPICS_DB_ZRSV, 33
    EPICS_DB_ZRVL, 34
EPICS_DB_ALLOWED
    Lists of EPICS record field names, 35
EPICS_DB_NUMVAL
    Lists of EPICS record field names, 35
EPICS_DB_ZRST
    EPICS record field names, 33
EPICS_DB_ZRSV
    EPICS record field names, 33
EPICS_DB_ZRVL
    EPICS record field names, 34
enum_map
    Classes for describing a structure element, 15
enum_pair
    Classes for describing a structure element, 15
enumtype
    Classes for describing a type, 16
epics_record_enumEnd
    DevTc, 39
epics_conversion
    EpicsTpy::epics_conversion, 122
epics_list_processing
    EpicsTpy::epics_list_processing, 135
epics_record_enum
    DevTc, 39
EpicsDbGen.cpp, 234
    main, 234
EpicsTpy, 40
EpicsTpy::epics_conversion, 120
    epics_conversion, 122
    to_epics, 122
EpicsTpy::epics_db_processing, 124
    operator(), 125
    process_field_alarm, 129
    process_field_numeric, 130, 131
    process_field_string, 132
EpicsTpy::epics_list_processing, 133
    epics_list_processing, 135
    operator(), 136
EpicsTpy::split_io_support, 183
    increment, 185
    operator=, 186
    split_io_support, 185
erase
    plc::BasePLC, 64
errorPrintf
    TcComms, 47
eventval
    DevTc, 39

find
    plc::BasePLC, 64
for_each
    plc::BasePLC, 64
functionblock
    Classes for describing a type, 16
Functions called by the EPICS base, 13

get
    ParseTpy::ads_routing_info, 50, 51
    ParseTpy::memory_location, 171
    ParseTpy::process_arg, 179
get_full
    ParseTpy::process_arg, 179
get_next
    plc::BasePLC, 65
get_timestamp_unix
    plc::BasePLC, 65
getopt
    ParseTpy::tag_processing, 193

histogramval
    DevTc, 39

increment
    EpicsTpy::split_io_support, 185
InfoInterface, 163
InfoPLC, 165
    ~InfoPLC, 166
    InfoPLC, 165
    InfoPLC, 165
infoPlc.h, 239
Init
    plc::DataValue, 87
Interface
    plc::Interface, 168
InterfacePtr
    plc, 46
invalidval
    DevTc, 39
iocMain.cpp, 239
isCallback
    DevTc::EpicsInterface, 161
isPassive
    DevTc::EpicsInterface, 161
item_list
    Classes for describing a structure element, 15

ligo_std
    Utility functions and classes, 30
linkRecord
    devTc.cpp, 226
listing_autoburt
    Classes for converting a paresed tpy, 31
listing_epics
    Classes for converting a paresed tpy, 31
listing_standard
    Classes for converting a paresed tpy, 31
listing_type
    Classes for converting a paresed tpy, 31
Lists of EPICS record field names, 35
    EPICS_DB_NUMVAL, 35
longinval
    DevTc, 39

longoutval
DevTc, 39
lower_case
Utility functions and classes, 30

main
EpicsDbGen.cpp, 234
ParseTpyInfo.cpp, 246
mbbiDirectrval
DevTc, 39
mbbiDirectval
DevTc, 39
mbbirval
DevTc, 39
mbbival
DevTc, 39
mbboDirectrval
DevTc, 39
mbboDirectval
DevTc, 39
mbborval
DevTc, 39
mbboval
DevTc, 39
mysize
plc::DataValue, 110

name_parse
ParseTpy::parserinfo_type, 176
no_change
OPC related functions and classes, 14
no_conversion
Utility functions and classes, 30
no_dot
Utility functions and classes, 30

OPC related functions and classes
no_change, 14
publish, 14
silent, 14
OPC property constants, 22
OPC related functions and classes, 14
opc_enum, 14
property_el, 14
property_map, 14
OPC tpy file constants, 21
opc_enum
OPC related functions and classes, 14
operator()
DevTc::epics_tc_db_processing, 157
EpicsTpy::epics_db_processing, 125
EpicsTpy::epics_list_processing, 136
std::std::hash< stringcase >, 162
std::std::hash< wstringcase >, 163
operator=
EpicsTpy::split_io_support, 186
std::atomic_string, 57
optimizeRequests
TcComms::TcPLC, 203

parse_finish
ParseTpy::tpy_file, 211
ParseTpy, 41
ParseTpy.cpp, 240
ParseTpy.h, 241
ParseTpy::ads_routing_info, 49
get, 50, 51
set, 51
ParseTpy::base_record, 57
base_record, 59, 60
ParseTpy::bit_location, 80
ParseTpy::item_record, 168
ParseTpy::memory_location, 169
get, 171
set, 171
set_section, 172
ParseTpy::opc_list, 173
ParseTpy::parserinfo_type, 174
name_parse, 176
ParseTpy::process_arg, 176
get, 179
get_full, 179
process_arg, 178
set, 180
ParseTpy::project_record, 181
ParseTpy::symbol_record, 187
ParseTpy::tag_processing, 190
getopt, 193
tag_processing, 192, 193
ParseTpy::tpy_file, 209
parse_finish, 211
process_array, 211
process_symbols, 213
process_type_tree, 214, 216, 218
ParseTpy::type_record, 220
ParseTpyConst.h, 243
ParseTpyInfo.cpp, 245
main, 246
ParseTpyTemplate.h, 248
plc, 44
BasePLCList, 45
BasePLCPtr, 45
BaseRecordList, 46
BaseRecordPtr, 46
data_type_enum, 46
dtBinary, 46
dtBool, 46
dtDouble, 46
dtFloat, 46
dtInt16, 46
dtInt32, 46
dtInt64, 46
dtInt8, 46
dtInvalid, 46
dtString, 46
dtUInt16, 46
dtUInt32, 46
dtUInt64, 46

dtUInt8, 46
dtWString, 46
InterfacePtr, 46
plc::BasePLC, 60
add, 63
erase, 64
find, 64
for_each, 64
get_next, 65
get_timestamp_unix, 65
records, 66
reserve, 66
plc::BaseRecord, 66
BaseRecord, 69
PlcPush, 69
PlcRead, 70, 71
PlcReadBinary, 72
PlcWrite, 72, 74
PlcWriteBinary, 74
UserPush, 75
UserRead, 76, 77
UserReadBinary, 77
UserWrite, 78, 79
UserWriteBinary, 79
plc::DataValue, 84
DataValue, 86
Init, 87
mysize, 110
PlcRead, 88, 90, 91
PlcReadBinary, 91
PlcWrite, 92–94
PlcWriteBinary, 94
Read, 95, 96, 98
ReadBinary, 99
UserRead, 100–102
UserReadBinary, 102
UserWrite, 103–105
UserWriteBinary, 105
Write, 106–108
WriteBinary, 109
plc::DataValueTraits< T >, 111
plc::DataValueTypeDef, 112
plc::Interface, 166
Interface, 168
plc::System, 189
add, 190
plc::scanner_thread_args, 182
plcBase.cpp, 248
plcBase.h, 249
plcBaseTemplate.h, 252
PlcPush
plc::BaseRecord, 69
PlcRead
plc::BaseRecord, 70, 71
plc::DataValue, 88, 90, 91
PlcReadBinary
plc::BaseRecord, 72
plc::DataValue, 91
PlcWrite
plc::BaseRecord, 72, 74
plc::DataValue, 92–94
PlcWriteBinary
plc::BaseRecord, 74
plc::DataValue, 94
preserve_case
Utility functions and classes, 30
printTCatVal
TcComms::TCatInterface, 199
process_all
Classes for describing the parser, 18
process_atomic
Classes for describing the parser, 18
process_structured
Classes for describing the parser, 18
process_arg
ParseTpy::process_arg, 178
process_array
ParseTpy::tpy_file, 211
process_field_alarm
EpicsTpy::epics_db_processing, 129
process_field_numeric
EpicsTpy::epics_db_processing, 130, 131
process_field_string
EpicsTpy::epics_db_processing, 132
process_symbols
ParseTpy::tpy_file, 213
process_tag_enum
Classes for describing the parser, 18
process_type_enum
Classes for describing the parser, 18
process_type_tree
ParseTpy::tpy_file, 214, 216, 218
property_el
OPC related functions and classes, 14
property_map
OPC related functions and classes, 14
pt_binary
Classes for describing the parser, 18
pt_bool
Classes for describing the parser, 18
pt_enum
Classes for describing the parser, 18
pt_int
Classes for describing the parser, 18
pt_invalid
Classes for describing the parser, 18
pt_real
Classes for describing the parser, 18
pt_string
Classes for describing the parser, 18
publish
OPC related functions and classes, 14
pull
DevTc::EpicsInterface, 161
push
DevTc::EpicsInterface, 161

Read
plc::DataValue, 95, 96, 98
read_ptr
TcComms::tcProcWrite, 208
ReadBinary
plc::DataValue, 99
records
plc::BasePLC, 66
req
TcComms::tcProcWrite, 208
reserve
plc::BasePLC, 66

set
ParseTpy::ads_routing_info, 51
ParseTpy::memory_location, 171
ParseTpy::process_arg, 180
set_addr
TcComms::TcPLC, 205
set_section
ParseTpy::memory_location, 172
silent
OPC related functions and classes, 14
simple
Classes for describing a type, 16
split_io_support
EpicsTpy::split_io_support, 185
std::atomic< string >, 53
std::atomic< wstring >, 55
std::atomic_string
operator=, 57
std::atomic_string< stringT >, 56
std::case_char_traits, 81
std::case_wchar_traits, 82
std::std::hash< stringcase >, 162
operator(), 162
std::std::hash< wstringcase >, 162
operator(), 163
stdafx.cpp, 253
stdafx.h, 253
stringcase
Utility functions and classes, 24
stringcase.cpp, 254
stringcase.h, 255
stringcase_hash.h, 256
stringinval
DevTc, 39
stringoutval
DevTc, 39
strncasecmp
Utility functions and classes, 25
structtype
Classes for describing a type, 16
symbol_list
Classes for describing a symbol, 17
syminfo_processing, 188

tCat_registerRecordDeviceDriver.cpp, 257
TCatInterface
TcComms::TCatInterface, 198
tag_processing
ParseTpy::tag_processing, 192, 193
tc_epics_conv
Utility functions and classes, 30
TcComms, 47
errorPrintf, 47
tcComms.cpp, 260
tcComms.h, 261
TcComms::DataPar, 83
TcComms::TCatInterface, 195
printTCatVal, 199
TCatInterface, 198
TcComms::TcPLC, 201
closePort, 203
optimizeRequests, 203
set_addr, 205
TcComms::tcProcWrite, 206
add, 207
read_ptr, 208
req, 208
to_epics
EpicsTpy::epics_conversion, 122
TpyToEpics.cpp, 262
TpyToEpics.h, 263
TpyToEpicsConst.h, 264
trim_space
Utility functions and classes, 25
type_enum
Classes for describing a type, 16
type_map
Classes for describing a type, 16

unknown
Classes for describing a type, 16
upper_case
Utility functions and classes, 30
UserPush
plc::BaseRecord, 75
UserRead
plc::BaseRecord, 76, 77
plc::DataValue, 100–102
UserReadBinary
plc::BaseRecord, 77
plc::DataValue, 102
UserWrite
plc::BaseRecord, 78, 79
plc::DataValue, 103–105
UserWriteBinary
plc::BaseRecord, 79
plc::DataValue, 105
Utility functions and classes, 24, 30
case_type, 30
ligo_std, 30
lower_case, 30
no_conversion, 30
no_dot, 30
preserve_case, 30
stringcase, 24

strncasecmp, 25
tc_epics_conv, 30
trim_space, 25
upper_case, 30
wcsncasewcmp, 26
wstringcase, 24

value_ait_type
DevTc::epics_record_traits, 138
value_count
DevTc::epics_record_traits, 138

waveformval
DevTc, 39
wcsncasewcmp
Utility functions and classes, 26
Write
plc::DataValue, 106–108
WriteBinary
plc::DataValue, 109
wstringcase
Utility functions and classes, 24

XML tpy file constants, 20